Vivado Design Suite Tcl Command Reference Guide

UG835 (v2021.1) June 16, 2021

XILINX®

Revision History

The following table shows the revision history for this document:

Section	Revision Summary
06/16/2021 Version 2021.1	
copy_constraints, create_single_pass_run, get_constant_paths, report_constant_paths, report_sim_version, upgrade_project, write_xsim_coverage	Commands Added in 2021.1
combine_hw_platforms, compile_simlib, config_ip_cache, create_cluster_configuration, create_hw_device, create_slack_histogram, create_waiver, export_simulation, find_routing_path, get_bels, implement_mig_cores, implement_xphy_cores, iphys_opt_design, place_design, read_iphys_opt_tcl, report_clock_interaction, report_design_analysis, report_exceptions, report_methodology, report_qor_assessment, report_qor_suggestions, save_bd_design_as, validate_bd_design, write_qor_suggestions, write_xdc, xsim	Commands Modified in 2021.1
create_rqs_run	Commands Removed in 2021.1

Introduction

Navigating Content by Design Process

Xilinx® documentation is organized around a set of standard design processes to help you find relevant content for your current development task. All Versal™ ACAP design process Design Hubs can be found on the Xilinx.com website. This document covers the following design processes:

- **Hardware, IP, and Platform Development:** Creating the PL IP blocks for the hardware platform, creating PL kernels, functional simulation, and evaluating the Vivado® timing, resource use, and power closure. Also involves developing the hardware platform for system integration.

- **System Integration and Validation:** Integrating and validating the system functional performance, including timing, resource use, and power closure.

- **Board System Design:** Designing a PCB through schematics and board layout. Also involves power, thermal, and signal integrity considerations.

Overview of Tcl Capabilities in Vivado

The Tool Command Language (Tcl) is the scripting language integrated in the Vivado® tool environment. Tcl is a standard language in the semiconductor industry for application programming interfaces, and is used by Synopsys® Design Constraints (SDC).

SDC is the mechanism for communicating timing constraints for FPGA synthesis tools from Synopsys Synplify as well as other vendors, and is a timing constraint industry standard; consequently, the Tcl infrastructure is a "Best Practice" for scripting language.

Tcl lets you perform interactive queries to design tools in addition to executing automated scripts. Tcl offers the ability to "ask" questions interactively of design databases, particularly around tool and design settings and state. Examples are: querying specific timing analysis reporting commands live, applying incremental constraints, and performing queries immediately after to verify expected behavior without re-running any tool steps.

The following sections describe some of the basic capabilities of Tcl with Vivado.

Note: This manual is not a comprehensive reference for the Tcl language. It is a reference to the specific capabilities of the Vivado Design Suite Tcl shell, and provides reference to additional Tcl programming resources.

Launching the Vivado Design Suite

You can launch the Vivado Design Suite and run the tools using different methods depending on your preference. For example, you can choose a Tcl script-based compilation style method in which you manage sources and the design process yourself, also known as Non-Project Mode. Alternatively, you can use a project-based method to automatically manage your design process and design data using projects and project states, also known as Project Mode. Either of these methods can be run using a Tcl scripted batch mode or run interactively in the Vivado IDE. For more information on the different design flow modes, see the *Vivado Design Suite User Guide: Design Flows Overview* (UG892).

Tcl Shell Mode

If you prefer to work directly with Tcl commands, you can interact with your design using Tcl commands with one of the following methods:

- Enter individual Tcl commands in the Vivado Design Suite Tcl shell outside of the Vivado IDE.

- Enter individual Tcl commands in the Tcl Console at the bottom of the Vivado IDE.

- Run Tcl scripts from the Vivado Design Suite Tcl shell.

- Run Tcl scripts from the Vivado IDE.

Use the following command to invoke the Vivado Design Suite Tcl shell either at the Linux command prompt or within a Windows Command Prompt window:

```
vivado -mode tcl
```

> **TIP:** *On Windows, you can also select **Start → All Programs → Xilinx Design Tools → Vivado yyyy.x → Vivado yyyy.x Tcl Shell**, where "yyyy.x" is the installed version of Vivado.*

For more information about using Tcl and Tcl scripting, see the *Vivado Design Suite User Guide: Using the Tcl Scripting Capabilities* (UG894). For a step-by-step tutorial that shows how to use Tcl in the Vivado tool, see the *Vivado Design Suite Tutorial: Design Flows Overview* (UG888).

Send Feedback

Tcl Batch Mode

You can use the Vivado tools in batch mode by supplying a Tcl script when invoking the tool. Use the following command either at the Linux command prompt or within a Windows Command Prompt window:

```
vivado -mode batch -source <your_Tcl_script>
```

The Vivado Design Suite Tcl shell will open, run the specified Tcl script, and exit when the script completes. In batch mode, you can queue up a series of Tcl scripts to process a number of designs overnight through synthesis, simulation, and implementation, and review the results on the following morning.

You can also pass arguments to the Vivado command when sourcing a Tcl script in batch mode. The `-tclargs` option lets you specify arguments for the Tcl script you are running. For example:

```
vivado -mode batch -source script.tcl -tclargs "FPGA=115-2"
```

IMPORTANT! *You must enclose the Tcl argument and value in quotes as shown in the example above, or there can be an error in handling the argument.*

Vivado IDE Mode

You can launch the Vivado Design Suite and run the tools using different methods depending on your preference. For example, you can choose a Tcl script-based compilation style method in which you manage sources and the design process yourself, also known as Non-Project Mode. Alternatively, you can use a project-based method to automatically manage your design process and design data using projects and project states, also known as Project Mode. Either of these methods can be run using a Tcl scripted batch mode or run interactively in the Vivado IDE. For more information on the different design flow modes, see the *Vivado Design Suite User Guide: Design Flows Overview* (UG892).

If you prefer to work in a GUI, you can launch the Vivado IDE from Windows or Linux. For more information on the Vivado IDE, see the *Vivado Design Suite User Guide: Using the Vivado IDE* (UG893).

Launch the Vivado IDE from your working directory. By default the Vivado journal and log files, and any generated report files, are written to the directory from which the Vivado tool is launched. This makes it easier to locate the project file, log files, and journal files, which are written to the launch directory.

In the Windows OS, select **Start→All Programs→Xilinx Design Tools→Vivado yyyy.x→Vivado yyyy.x Tcl Shell**, where "yyyy.x" is the installed version of Vivado.

TIP: *You can also double-click the Vivado IDE shortcut icon on your Windows desktop.*

Send Feedback

In the Linux OS, enter the following command at the command prompt:

```
vivado -or- vivado -mode gui
```

If you need help, with the Vivado tool command line executable, type:

```
vivado -help
```

If you are running the Vivado tool from the Vivado Design Suite Tcl shell, you can open the Vivado IDE directly from the Tcl shell by using the `start_gui` command.

From the Vivado IDE, you can close the Vivado IDE and return to a Vivado Tcl shell by using the `stop_gui` command.

Tcl Journal Files

When you invoke the Vivado tool, it writes the `vivado.log` file to record the various commands and operations performed during the design session. The Vivado tool also writes a file called `vivado.jou` which is a journal of just the Tcl commands run during the session. The journal file can be used as a source to create new Tcl scripts.

Note: Backup versions of the journal file, named `vivado_<id>.backup.jou`, are written to save the details of prior runs whenever the Vivado tool is launched. The <id> is a unique identifier that allow the tool to create and store multiple backup versions of the log and journal files.

Tcl Help

The Tcl `help` command provides information related to the supported Tcl commands.

• `help` – Returns a list of Tcl command categories.

  ```
  help
  ```

  Command categories are groups of commands performing a specific function, like File I/O for instance.

• `help -category` *category* – Returns a list of commands found in the specified category.

  ```
  help -category object
  ```

  This example returns the list of Tcl commands for handling objects.

• `help` *pattern* – Returns a list of commands that match the specified search pattern. This form can be used to quickly locate a specific command from a group of commands.

  ```
  help get_*
  ```

Send Feedback

This example returns the list of Tcl commands beginning with get_.

- `help` *command* – Provides detailed information related to the specified command.

  `help get_cells`

  This example returns specific information of the get_cells command.

- `help -args` *command* – Provides an abbreviated help text for the specified command, including the command syntax and a brief description of each argument.

  `help -args get_cells`

- `help -syntax` *command* – Reports the command syntax for the specified command.

  `help -syntax get_cells`

Scripting in Tcl

Tcl Initialization Scripts

> **TIP:** *The following describes where you can place* `Vivado_init.tcl` *scripts if you would like to customize Vivado on startup. No* `Vivado_init.tcl` *scripts are provided in the Vivado release by default.*

When you start the Vivado tool, it looks for a Tcl initialization script in three different locations, each one overriding the last one found:

1. Enterprise: In the software installation directory, `<installdir>/Vivado/<version>/scripts/Vivado_init.tcl`

2. Vivado Version: In a local user directory, for a specific version of the Vivado Design Suite:
   - For Windows 7: `%APPDATA%/Xilinx/Vivado/<version>/Vivado_init.tcl`
   - For Linux: `$HOME/.Xilinx/Vivado/<version>/Vivado_init.tcl`

3. Vivado User: In a local user directory, for the general Vivado Design Suite:
   - For Windows 7: `%APPDATA%/Xilinx/Vivado/Vivado_init.tcl`
   - For Linux: `$HOME/.Xilinx/Vivado/Vivado_init.tcl`

Where:

- <installdir> is the installation directory where the Vivado Design Suite is installed.

If `Vivado_init.tcl` exists, in one or all of these locations, the Vivado tool sources this file, in the order described above.

- The `Vivado_init.tcl` file in the installation directory allows a company or design group to support a common initialization script for all users. Anyone starting the Vivado tool from that installation location sources the enterprise `Vivado_init.tcl` script.

- A user's `Vivado_init.tcl` file in the home directory allows each user to specify additional commands, or to override commands from the software installation to meet their specific design requirements.

- No `Vivado_init.tcl` file is provided with the Vivado Design Suite installation. You must create the `Vivado_init.tcl` file and place it in either the installation directory, or your home directory, as discussed to meet your specific needs.

> **TIP:** *Other tools in the Vivado Design Suite also support initialization scripts in the following form: <tool>_init.tcl, where <tool> can include Vivado, vivado_lab, xsim, and xelab.*

The `Vivado_init.tcl` file is a standard Tcl command file that can contain any valid Tcl command supported by the Vivado tool. You can also source another Tcl script file from within `Vivado_init.tcl` by adding the following statement:

```
source <path_to_file>/<file_name>.tcl
```

Note: You can also specify the `-init` option when launching the Vivado Design Suite from the command line. Type `vivado -help` for more information.

Sourcing a Tcl Script

A Tcl script can be sourced from either one of the command-line options or from the GUI. Within the Vivado Integrated Design Environment (IDE) you can source a Tcl script from **Tools → Run Tcl Script**.

You can source a Tcl script from a Tcl command-line option:

```
source <file_name>
```

When you invoke a Tcl script from the Vivado IDE, a progress bar is displayed and all operations in the IDE are blocked until the scripts completes.

There is no way to interrupt script execution during run time; consequently, standard OS methods of killing a process must be used to force interruption of the tool. If the process is killed, you lose any work done since your last save.

Typing `help source` in the Tcl console will provide additional information regarding the `source` command.

Using Tcl.pre and Tcl.post Hook Scripts

Tcl Hook scripts allow you to run custom Tcl scripts prior to (`tcl.pre`) and after (`tcl.post`) synthesis and implementation design runs, or any of the implementation steps. Whenever you launch a run, the Vivado tool uses a predefined Tcl script which executes a design flow based on the selected strategy. Tcl Hook scripts let you customize the standard flow, with pre-processors or post-processors, such as for generating custom reports. The Tcl Hook script must be a standard Tcl script.

Every step in the design flow has a pre- and post-hook capability. Common examples are:

- Custom reports: timing, power, utilization, or any user-defined tcl report.
- Temporary parameters for workarounds.
- Over-constraining timing constraints for portions of the flow.
- Multiple iterations of stages (e.g. multiple calls to phys_opt_design).
- Modifications to netlist, constraint, or device programming.

> **IMPORTANT!** *Relative paths within the tcl.pre and tcl.post scripts are relative to the appropriate run directory of the project they are applied to:* `<project>/<project.runs>/<run_name>`. *You can use the DIRECTORY property of the current project or current run to define the relative paths in your Tcl hook scripts:*

```
get_property DIRECTORY [current_project] get_property DIRECTORY
[current_run]
```

For more information on defining Tcl Hook scripts, refer to the *Vivado Design Suite User Guide: Using Tcl Scripting* (UG894).

General Tcl Syntax Guidelines

Tcl uses the Linux file separator (/) convention regardless of which Operating System you are running.

The following subsections describe the general syntax guidelines for using Tcl in the Vivado Design Suite.

Using Tcl Eval

When executing Tcl commands, you can use variable substitution to replace some of the command line arguments accepted or required by the Tcl command. However, you must use the Tcl `eval` command to evaluate the command line with the Tcl variable as part of the command.

For instance, the `help` command can take the `-category` argument, with one of a number of command categories as options:

```
help -category ipflow
```

You can define a variable to hold the command category:

```
set cat "ipflow"
```

Where:

- `set` is the Tcl keyword that defines the variable.

- `cat` is the name of the variable being defined.

- `"ipflow"` is the value assigned to the variable.

You can then evaluate the variable in the context of the Tcl command:

```
eval help -category $cat
```

or,

```
set cat "category ipflow" eval help $cat
```

You can also use braces {} in place of quotation marks "" to achieve the same result:

```
set runblocksOptDesignOpts { -sweep -retarget -propconst -remap }
eval opt_design $runblocksOptDesignOpts
```

Typing `help eval` in the Tcl console will provide additional information regarding the `eval` command.

Using Special Characters

Some commands take arguments that contain characters that have special meaning to Tcl. Those arguments must be surrounded with curly braces {} to avoid unintended processing by Tcl. The most common cases are as follows.

Bus Indexes - Because square brackets `[ ]` have special meaning to Tcl, an indexed (bit- or part-selected) bus using the square bracket notation must be surrounded with curly braces. For example, when adding index 4 of a bus to the Vivado Common Waveform Viewer window using the square bracket notation, you must write the command as:

```
add_wave {bus[4]}
```

Parentheses can also be used for indexing a bus, and because parentheses have no special meaning to Tcl, the command can be written without curly braces. For example:

```
add_wave bus(4)
```

Send Feedback

Verilog Escaped Identifiers - Verilog identifiers containing characters or keywords that are reserved by Verilog need to be "escaped" both in the Verilog source code and on the simulator command line by prefixing the identifier with a backslash "\" and appending a space. Additionally, on the Tcl command line the escaped identifier must be surrounded with curly braces.

Note: If an identifier already includes a curly brace, then the technique of surrounding the identifier with curly braces does not work, because Tcl interprets curly braces as reserved characters even nested within curly braces. Instead, you must use the technique described below, in **VHDL Extended Identifiers**.

For example, to add a wire named "`my wire`" to the Vivado Common Waveform Viewer window, you must write the command as:

```
add_wave {\my wire }
```

Note: Be sure to append a space after the final character, and before the closing brace.

Verilog allows any identifier to be escaped. However, on the Tcl command line do not escape identifiers that are not required to be escaped. For example, to add a wire named "w" to the Vivado Common Waveform Viewer window, the Vivado simulator would not accept:

```
add_wave {\w }
```

as a valid command, since this identifier (the wire name "w") does not required to be escaped. The command must be written as:

```
add_wave w
```

VHDL Extended Identifiers - VHDL extended identifiers contain backslashes, "\", which are reserved characters in Tcl. Because Tcl interprets a backslash next to a close curly brace \} as being a close curly brace character, VHDL extended identifiers cannot be written with curly braces. Instead, the curly braces must be absent and each special character to Tcl must be prefixed with a backslash. For example, to add the signal \my sig\ to the Wave window, you must write the command as:

```
add_wave \\my\ sig\\
```

Note: Both the backslashes that are part of the extended identifier, and the space inside the identifier are prefixed with a backslash.

General Syntax Structure

The general structure of Vivado Design Suite Tcl commands is:

```
command [optional_parameters] required_parameters
```

Command syntax is of the verb-noun and verb-adjective-noun structure separated by the underscore ("_") character.

Commands are grouped together with common prefixes when they are related.

- Commands that query things are generally prefixed with `get_`.

- Commands that set a value or a parameter are prefixed with `set_`.

- Commands that generate reports are prefixed with `report_`.

The commands are exposed in the global namespace. Commands are "flattened," meaning there are no "sub-commands" for a command.

Example Syntax

The following example shows the return format on the `get_cells -help` command:

```
get_cells

Description:
Get a list of cells in the current design

Syntax:
get_cells  [-hsc <arg>] [-hierarchical] [-regexp] [-nocase] [-filter <arg>]
           [-of_objects <args>] [-match_style <arg>] [-quiet] [-verbose]
           [<patterns>]

Returns:
list of cell objects

Usage:
  Name               Description
  -------------------------------
  [-hsc]             Hierarchy separator
                     Default: /
  [-hierarchical]    Search level-by-level in current instance
  [-regexp]          Patterns are full regular expressions
  [-nocase]          Perform case-insensitive matching (valid only when -
regexp
                     specified)
  [-filter]          Filter list with expression
  [-of_objects]      Get cells of these pins, timing paths, nets, bels, sites
                     or drc violations
  [-match_style]     Style of pattern matching
                     Default: sdc
                     Values: ucf, sdc
  [-quiet]           Ignore command errors
  [-verbose]         Suspend message limits during command execution
  [<patterns>]       Match cell names against patterns
                     Default: *

Categories:
SDC, XDC, Object
```

Unknown Commands

Tcl contains a list of built-in commands that are generally supported by the language, Vivado tool specific commands which are exposed to the Tcl interpreter, and user-defined procedures.

Send Feedback

Commands that do not match any of these known commands are sent to the OS for execution in the shell from the `exec` command. This lets users execute shell commands that might be OS-specific. If there is no shell command, then an error message is issued to indicate that no command was found.

Return Codes

Some Tcl commands are expected to provide a return value, such as a list or collection of objects on which to operate. Other commands perform an action but do not necessarily return a value that can be used directly by the user. Some tools that integrate Tcl interfaces return a 0 or a 1 to indicate success or error conditions when the command is run.

To properly handle errors in Tcl commands or scripts, you should use the Tcl built-in command `catch`. Generally, the `catch` command and the presence of numbered info, warning, or error messages should be relied upon to assess issues in Tcl scripted flows.

Vivado tool Tcl commands return either TCL_OK or TCL_ERROR upon completion. In addition, the Vivado Design Suite sets the global variable $ERRORINFO through standard Tcl mechanisms.

To take advantage of the $ERRORINFO variable, use the following line to report the variable after an error occurs in the Tcl console:

```
puts $ERRORINFO
```

This reports specific information to the standard display about the error. For example, the following code example shows a Tcl script (`procs.tcl`) being sourced, and a user-defined procedure (loads) being run. There are a few transcript messages, and then an error is encountered at line 5.

```
Line 1: Vivado % source procs.tcl
Line 2: Vivado% loads
Line 3: Found 180 driving FFs
Line 4: Processing pin a_reg_reg[1]/Q...
Line 5: ERROR: [HD-Tcl 53] Cannot specify '-patterns' with '-of_objects'.
Line 6: Vivado% puts $errorInfo
Line 7: ERROR: [HD-Tcl 53] Cannot specify '-patterns' with '-of_objects'.
    While executing "get_ports -of objects $pin" (procedure "my_report" line 6)
    invoked from within procs.tcl
```

You can add `puts $ERRORINFO` into catch clauses in your Tcl script files to report the details of an error when it is caught, or use the command interactively in the Tcl console immediately after an error is encountered to get the specific details of the error.

In the example code above, typing the `puts $ERRORINFO` command in line 6, reports detailed information about the command and its failure in line 7.

First Class Tcl Objects and Relationships

The Tcl commands in the Vivado Design Suite provide direct access to the object models for netlist, devices, and projects. These are Vivado first-class objects, which means they are more than just a string representation, and they can be operated on and queried. There are a few exceptions to this rule, but generally "things" can be queried as objects, and these objects have properties that can be queried and they have relationships that allow you to get to other objects.

Object Types and Definitions

There are many object types in the Vivado Design Suite; this chapter provides definitions and explanations of the basic types. The most basic and important object types are associated with entities in a design netlist, and these types are listed in the following subsections:

- **Cell:** A cell is an instance, either primitive or hierarchical inside a netlist. Examples of cells include flip-flops, LUTs, I/O buffers, RAM and DSPs, as well as hierarchical instances which are wrappers for other groups of cells.

- **Pin:** A pin is a point of logical connectivity on a cell. A pin allows the internals of a cell to be abstracted away and simplified for easier use, and can either be on hierarchical or primitive cells. Examples of pins include clock, data, reset, and output pins of a flop.

- **Port:** A port is a connection at an object boundary used to connect internal content to the outside of the object. Ports in the top-level netlist or design are normally attached to I/O pads on the die, connected to pins on the device package, and connected externally to the device in a system-level design. Ports inside of a hierarchical cell, module, or entity, are represented as pins on the hierarchical cell.

- **Net:** A net is a wire or list of wires that eventually be physically connected directly together. Nets can be hierarchical or flat, but always sorts a list of pins together.

- **Clock:** A clock is a periodic signal that propagates to sequential logic within a design. Clocks can be primary clock domains or generated by clock primitives such as a DCM, PLL, or MMCM. A clock is the rough equivalent to a TIMESPEC PERIOD constraint in UCF and forms the basis of static timing analysis algorithms.

Querying Objects

All first class objects can be queried by a `get_*` Tcl command that generally has the following syntax:

- `get_<object_type> <pattern>`

Send Feedback

Where pattern is a search <pattern>, which includes if applicable a hierarchy separator to get a fully qualified name. Objects are generally queried by a string pattern match applied at each level of the hierarchy, and the search pattern also supports wildcard style search patterns to make it easier to find objects, for example:

- `get_cells */inst_1`

This command searches for a cell named inst_1 within the first level of hierarchy under the top-level of hierarchy. To recursively search for a pattern at every level of hierarchy, use the following syntax:

- `get_cells -hierarchical inst_1`

This command searches every level of hierarchy for any instances that match inst_1.

For complete coverage of the command syntax, see the specific online help for the individual command:

- `help get_cells`

- `get_cells -help`

Object Properties

Objects have properties that can be queried. Property names are unique for any given object type. To query a specific property for an object, the following command is provided:

- `get_property <property_name> <object>`

For example, the lib_cell property on cell objects tells you what UniSim component a given instance is mapped to:

- `get_property lib_cell [get_cell inst_1]`

To discover all of the available properties for a given object type, use the `report_property` command:

- `report_property [get_cells inst_1]`

The following table shows the properties returned for a specific object.

Key	Value	Type
bel	OLOGICE1.OUTFF	string
class	cell	string
iob	TRUE	string
is_blackbox	0	bool
is_fixed	0	bool
is_partition	0	bool

Key	Value	Type
is_primitive	1	bool
is_reconfigurable	0	bool
is_sequential	1	bool
lib_cell	FD	string
loc	OLOGIC_X1Y27	string
name	error	string
primitive_group	FD_LD	string
primitive_subgroup	flop	string
site	OLOGIC_X1Y27	string
type	FD & LD	string
XSTLIB	1	bool

Some properties are read-only and some are user-settable. Properties that map to attributes that can be annotated in UCF or in HDL are generally user-settable through Tcl with the `set_property` command:

- `set_property loc OLOGIC_X1Y27 [get_cell inst_1]`

Filtering Based on Properties

The object query `get_*` commands have a common option to filter the query based on any property value attached to the object. This is a powerful capability for the object query commands. For example, to query all cells of primitive type FD do the following:

- `get_cells * -hierarchical -filter "lib_cell == FD"`

To do more elaborate string filtering, utilize the =~ operator to do string pattern matching. For example, to query all flip-flop types in the design, do the following:

- `get_cells * -hierarchical -filter "lib_cell =~ FD*"`

Multiple filter properties can be combined with other property filters with logical OR (||) and AND (&&) operators to make very powerful searches. To query every cell in the design that if of any flop type and has a placed location constraint:

- `get_cells * -hierarchical -filter {lib_cell =~ FD* && loc != ""}`

Note: In the example, the filter option value was wrapped with curly braces {} instead of double quotes. This is normal Tcl syntax that prevents command substitution by the interpreter and allows users to pass the empty string ("") to the loc property.

Handling Lists of Objects

Commands that return more than one object, such as `get_cells` or `get_sites`, return a collection in the Vivado tool that looks and behaves like a native Tcl list. This feature allows performance gains when handling large lists of Tcl objects without the need to use special commands like the `foreach_in_collection` command. In the Vivado Design Suite collections can be processed like Tcl lists using built-in commands such as `lsort`, `lsearch`, and `foreach`.

Typically, when you run a `get_*` command, the returned results are echoed to the console and to the log file as a Tcl string, rather than as a list due to a feature of Tcl called "shimmering". Internally, Tcl can store a variable or value both as a string and as a faster native object such as a float or a list object. In Tcl, shimmering occurs when the representation of the object or value changes from the list object to the string object, or from string to list. A list of Vivado objects is returned by the `get_*` command, but the shimmered string representation is written to the log file and Tcl console.

However, to improve performance and prevent overloading memory buffers, the Vivado Design Suite limits and truncates the shimmered string to a default character length defined by the `tcl.collectionResultDisplayLimit` parameter, which has a default value of 500. Commands that can return a significant number of objects, such as `get_cells` or `get_sites`, will truncate the returned string, ending it with an ellipsis ('...'). You can use the `set_param` command to change the `tcl.collectionResultDisplayLimit` parameter value to return more or fewer results.

> ⚠️ **CAUTION!** *The combination of shimmering and the* `tcl.collectionResultDisplayLimit` *parameter prevents the use of* `in` *and* `ni` *list operators in the Vivado Design Suite. Since a string shimmered from the list may be truncated, the* `in` *and* `ni` *operators cannot effectively determine if a specified object is in, or not-in, a list of objects. You should use list commands such as* `lsearch` *and* `lsort` *instead of* `in` *or* `ni`.

```
if {[lsearch -exact [get_cells *] $cellName] != -1} {...}
```

You can capture the complete list returned by the `get_*` command by assigning the results to a Tcl variable:

```
set allSites [get_sites]
```

The actual list in the variable assignment includes the complete result set, and is not truncated by the `tcl.collectionResultDisplayLimit` parameter. An example of this is seen in hierarchically querying all the cells in a design:

```
%set allCells [get_cells -hierarchical]
DataIn_pad_0_i_IBUF[0]_inst DataIn_pad_0_i_IBUF[1]_inst \
DataIn_pad_0_i_IBUF[2]_inst DataIn_pad_0_i_IBUF[3]_inst \
DataIn_pad_0_i_IBUF[4]_inst ...
%llength $allCells
42244
%lindex $allCells end
wbArbEngine/s4/next_reg
```

In the preceding example, the result of the hierarchical `get_cells` command was assigned to the `$allCells` variable. In appearance, the results are truncated. However, a check of the length of the list reports more than forty thousand cell objects, and the last index in the list returns an actual object, and not an ellipsis.

TIP: *If necessary, you can also use the `join` command, to join the list of objects returned by the `get_*` Tcl command, with a newline (\n), tab (\t), or a space (" "), to display the un-truncated list of objects:*

```
join [get_parts] " "
```

Object Relationships

Related objects can be queried using the `-of` option to the relevant `get_*` command. For example, to get a list of pins connected to a cell object, do the following:

* `get_pins -of [get_cells inst_1]`

The following image shows object types in the Vivado tool and their relationships, where an arrow from one object to another object indicates that you can use the `-of` option to the `get_*` command to traverse logical connectivity and get Tcl references to any connected object. For more information on first class objects and their relationships, refer to the *Vivado Design Suite Properties Reference Guide* (UG912).
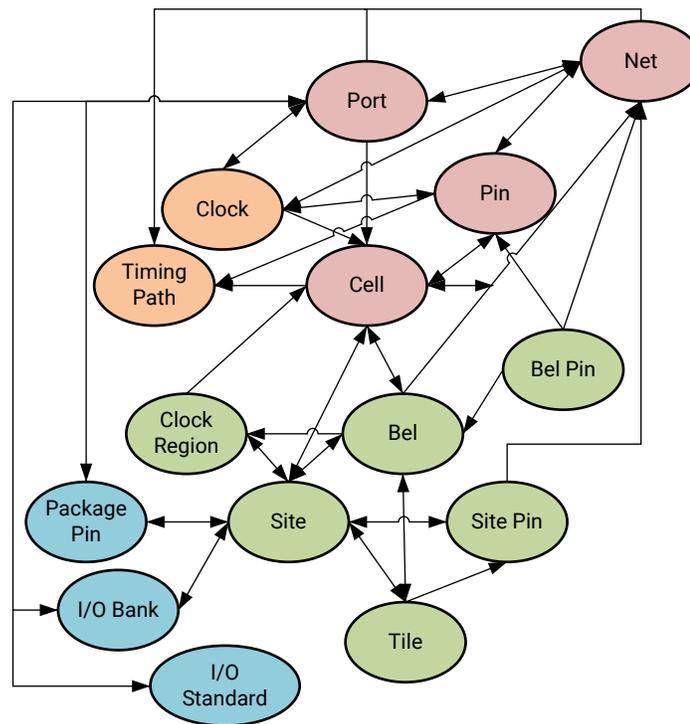
X25296-040621

Errors, Warnings, Critical Warnings, and Info Messages

Messages that result from individual commands appear in the log file as well as in the GUI console if it is active. These messages are generally numbered to identify specific issues and are prefixed in the log file with "INFO", "WARNING", "CRITICAL_Warning", "ERROR" followed by a subsystem identifier and a unique number.

The following example shows an INFO message that appears after reading the timing library.

```
INFO: [HD-LIB 1] Done reading timing library
```

These messages make it easier to search for specific issues in the log file to help to understand the context of operations during command execution.

Generally, when an error occurs in a Tcl command sourced from a Tcl script, further execution of subsequent commands is halted. This is to prevent unrecoverable error conditions. There are Tcl built-ins that allow users to intercept these error conditions, and to choose to continue. Consult any Tcl reference for the `catch` command for a description of how to handle errors using general Tcl mechanisms.

Tcl Commands

Tcl Command Categories

Board	cluster_configuration	Configuration
CreatePeripheral	Debug	DRC
Feasibility	FileIO	Floorplan
GUIControl	Hardware	IPFlow
IPIntegrator	Lint	Memory
Methodology	Netlist	Object
Partition	PinPlanning	Platform
Power	Project	projutils
PropertyAndParameter	Report	SDC
Simulation	SysGen	Timing
ToolLaunch	Tools	Vitis
vivado_preferences	Waiver	Waveform
XDC	xilinxtclstore	

Tcl Commands Listed by Category

Board:

apply_board_connection	current_board	current_board_part
get_board_bus_nets	get_board_buses	get_board_component_interfaces
get_board_component_modes	get_board_component_pins	get_board_components
get_board_interface_ports	get_board_ip_preferences	get_board_jumpers
get_board_parameters	get_board_part_interfaces	get_board_part_pins
get_board_parts	get_boards	validate_board_files

cluster_configuration:

create_cluster_configuration	get_cluster_configurations	remove_cluster_configurations
validate_cluster_configurations		

Configuration:

config_implementation

CreatePeripheral:

add_peripheral_interface   create_peripheral   generate_peripheral
write_peripheral

Debug:

apply_hw_ila_trigger   connect_debug_cores   connect_debug_port
create_debug_core   create_debug_port   delete_debug_core
delete_debug_port   disconnect_debug_port   get_debug_cores
get_debug_ports   implement_debug_core   modify_debug_ports
report_debug_core   write_debug_probes

DRC:

add_drc_checks   create_drc_check   create_drc_ruledeck
create_drc_violation   create_waiver   delete_drc_check
delete_drc_ruledeck   get_drc_checks   get_drc_ruledecks
get_drc_violations   remove_drc_checks   report_drc
reset_drc   reset_drc_check

Feasibility:

delete_qor_suggestions   get_qor_suggestions   read_qor_suggestions
report_qor_assessment   report_qor_suggestions   write_qor_suggestions

FileIO:

auto_detect_xpm   combine_hw_platforms   config_webtalk
create_port_on_reconfigurable_module decrypt_bitstream   encrypt
generate_base_platform   generate_mem_files   generate_pblock
generate_rl_platform   generate_shx_platform   infer_diff_pairs
open_hw_platform   pr_recombine   pr_subdivide
pr_verify   read_bd   read_checkpoint
read_csv   read_edif   read_ip
read_mem   read_qor_suggestions   read_saif
read_schematic   read_twx   read_verilog
read_vhdl   read_xdc   refresh_meminit
write_abstract_shell   write_bd_layout   write_bitstream
write_bmm   write_bsdl   write_cfgmem
write_checkpoint   write_csv   write_debug_probes
write_device_image   write_edif   write_hw_platform

Send Feedback

write_hw_platform_metadata
write_mem_info
write_sdf
write_xdc

write_ibis
write_qor_suggestions
write_verilog

write_inferred_xdc
write_schematic
write_vhdl

Floorplan:

add_cells_to_pblock
create_power_rail
delete_rpm
place_cell
resize_pblock

add_to_power_rail
delete_pblocks
get_pblocks
remove_cells_from_pblock
swap_locs

create_pblock
delete_power_rails
get_power_rails
remove_from_power_rail
unplace_cell

GUIControl:

create_gui_custom_command
get_gui_custom_command_args
get_marked_objects
mark_objects
remove_gui_custom_commands
show_schematic
stop_gui
unmark_objects

create_gui_custom_command_arg
get_gui_custom_commands
get_selected_objects
redo
select_objects
start_gui
undo
unselect_objects

endgroup
get_highlighted_objects
get_marked_objects
highlight_objects
remove_gui_custom_command_args
show_objects
startgroup
unhighlight_objects

Hardware:

add_hw_hbm_pc
close_hw_manager
commit_hw_mig
commit_hw_vio
create_hw_axi_txn
create_hw_device
create_hw_sio_linkgroup
create_hw_target
current_hw_ila
current_hw_target
delete_hw_cfgmem
detect_hw_sio_links
display_hw_sio_scan
get_hw_axi_txns
get_hw_ddrmcs
get_hw_ila_datas
get_hw_pcies
get_hw_sio_commons
get_hw_sio_iberts

add_hw_probe_enum
close_hw_target
commit_hw_sio
config_hw_sio_gts
create_hw_bitstream
create_hw_probe
create_hw_sio_scan
current_hw_cfgmem
current_hw_ila_data
delete_hw_axi_txn
delete_hw_probe
disconnect_hw_server
execute_hw_svf
get_hw_axis
get_hw_devices
get_hw_ilas
get_hw_probes
get_hw_sio_gtgroups
get_hw_sio_linkgroups

boot_hw_device
commit_hw_hbm
commit_hw_sysmon
connect_hw_server
create_hw_cfgmem
create_hw_sio_link
create_hw_sio_sweep
current_hw_device
current_hw_server
delete_hw_bitstream
delete_hw_target
display_hw_ila_data
get_cfgmem_parts
get_hw_cfgmems
get_hw_hbms
get_hw_migs
get_hw_servers
get_hw_sio_gts
get_hw_sio_links

get_hw_sio_plls

get_hw_sio_sweeps

get_hw_sysmon_reg

get_hw_vios

open_hw_target

program_hw_devices

read_hw_sio_sweep

refresh_hw_axi

refresh_hw_hbm

refresh_hw_server

refresh_hw_sysmon

remove_hw_hbm_pc

remove_hw_sio_linkgroup

report_hw_axi_txn

report_hw_pcie

reset_hw_axi

reset_hw_vio_activity

run_hw_axi

run_hw_sio_scan

runtest_hw_jtag

set_hw_sysmon_reg

stop_hw_sio_sweep

upload_hw_ila_data

wait_on_hw_sio_scan

write_hw_sio_scan

get_hw_sio_rxs

get_hw_sio_txs

get_hw_sysmons

list_hw_samples

pause_hw_hbm_amon

read_hw_ila_data

readback_hw_cfgmem

refresh_hw_ddrmc

refresh_hw_mig

refresh_hw_sio

refresh_hw_target

remove_hw_probe_enum

remove_hw_sio_scan

report_hw_ddrmc

report_hw_softmc

reset_hw_ila

reset_hw_vio_outputs

run_hw_hbm_amon

run_hw_sio_sweep

scan_dr_hw_jtag

stop_hw_hbm_amon

update_hw_firmware

verify_hw_devices

wait_on_hw_sio_sweep

write_hw_sio_sweep

get_hw_sio_scans

get_hw_softmcs

get_hw_targets

open_hw_manager

program_hw_cfgmem

read_hw_sio_scan

readback_hw_device

refresh_hw_device

refresh_hw_pcie

refresh_hw_softmc

refresh_hw_vio

remove_hw_sio_link

remove_hw_sio_sweep

report_hw_mig

report_hw_targets

reset_hw_pcie

resume_hw_hbm_amon

run_hw_ila

run_state_hw_jtag

scan_ir_hw_jtag

stop_hw_sio_scan

update_hw_gpio

wait_on_hw_ila

write_hw_ila_data

write_hw_svf

IPFlow:

add_peripheral_interface

convert_ips

create_ip_run

extract_files

get_ip_upgrade_results

import_ip

report_ip_status

update_ip_catalog

validate_ip

compile_c

copy_ip

create_peripheral

generate_peripheral

get_ipdefs

open_example_project

reset_target

update_module_reference

write_ip_tcl

config_ip_cache

create_ip

delete_ip_run

generate_target

get_ips

read_ip

synth_ip

upgrade_ip

write_peripheral

IPIntegrator:

apply_bd_automation

close_bd_design

connect_bd_net

create_bd_cell

create_bd_intf_pin

apply_board_connection

compile_c

copy_bd_objs

create_bd_design

create_bd_intf_port

assign_bd_address

connect_bd_intf_net

create_bd_addr_seg

create_bd_intf_net

create_bd_intf_tlm_port

create_bd_net   create_bd_pin   create_bd_port
create_bd_tlm_port   current_bd_design   current_bd_instance
delete_bd_objs   disconnect_bd_intf_net   disconnect_bd_net
exclude_bd_addr_seg   export_as_example_design   find_bd_objs
generate_target   get_bd_addr_segs   get_bd_addr_spaces
get_bd_cells   get_bd_designs   get_bd_intf_nets
get_bd_intf_pins   get_bd_intf_ports   get_bd_nets
get_bd_pins   get_bd_ports   get_bd_regs
get_example_designs   get_template_bd_designs   group_bd_cells
include_bd_addr_seg   instantiate_example_design instantiate_template_bd_design
make_bd_intf_pins_external make_bd_pins_external   move_bd_cells
open_bd_design   read_bd   regenerate_bd_layout
replace_bd_cell   report_bd_diffs   save_bd_design
save_bd_design_as   ungroup_bd_cells   update_bd_boundaries
upgrade_bd_cells   validate_bd_design   write_bd_tcl

Lint:

create_waiver

Memory:

implement_mig_cores   implement_xphy_cores   refresh_meminit

Methodology:

create_waiver   get_methodology_checks   get_methodology_violations
report_methodology   reset_methodology   reset_methodology_check

Netlist:

connect_net   create_cell   create_net
create_pin   disconnect_net   get_net_delays
remove_cell   remove_net   remove_pin
rename_cell   rename_net   rename_pin
rename_port   rename_ref   resize_net_bus
resize_pin_bus   tie_unused_pins

Object:

add_drc_checks   apply_board_connection   can_resolve_reference
config_ip_cache   create_drc_check   create_drc_ruledeck
create_partition_def   create_pr_configuration   create_reconfig_module
create_report_config   create_waiver   current_board
current_board_part   current_pr_configuration   delete_drc_check

Send Feedback

delete_drc_ruledeck

delete_report_configs

find_routing_path

get_bels

get_board_component_interfaces

get_board_components

get_board_jumpers

get_board_part_pins

get_cdc_violations

get_clock_regions

get_dashboard_gadgets

get_designs

get_drc_violations

get_generated_clocks

get_hw_axis

get_hw_devices

get_hw_ilas

get_hw_probes

get_hw_sio_gtgroups

get_hw_sio_linkgroups

get_hw_sio_rxs

get_hw_sio_txs

get_hw_targets

get_io_standards

get_ipdefs

get_lib_pins

get_marked_objects

get_net_delays

get_package_pins

get_path_groups

get_pips

get_ports

get_projects

get_reconfig_modules

get_selected_objects

get_sites

get_tiles

get_waivers

list_property

report_property

reset_drc_check

run_state_hw_jtag

scan_ir_hw_jtag

write_ip_tcl

delete_hw_bitstream

delete_waivers

generate_reports

get_board_bus_nets

get_board_component_modes

get_board_interface_ports

get_board_parameters

get_board_parts

get_cells

get_clocks

get_debug_cores

get_drc_checks

get_files

get_highlighted_objects

get_hw_cfgmems

get_hw_hbms

get_hw_migs

get_hw_servers

get_hw_sio_gts

get_hw_sio_links

get_hw_sio_scans

get_hw_softmcs

get_hw_vios

get_iobanks

get_ips

get_libs

get_methodology_checks

get_nets

get_partition_defs

get_pblocks

get_pkgpin_bytegroups

get_pr_configurations

get_property

get_report_configs

get_site_pins

get_slrs

get_timing_arcs

get_wires

list_property_value

report_qor_suggestions

reset_methodology_check

runtest_hw_jtag

set_property

write_waivers

delete_qor_suggestions

filter

get_bel_pins

get_board_buses

get_board_component_pins

get_board_ip_preferences

get_board_part_interfaces

get_boards

get_cfgmem_parts

get_constant_paths

get_debug_ports

get_drc_ruledecks

get_filesets

get_hw_axi_txns

get_hw_ddrmcs

get_hw_ila_datas

get_hw_pcies

get_hw_sio_commons

get_hw_sio_iberts

get_hw_sio_plls

get_hw_sio_sweeps

get_hw_sysmons

get_interfaces

get_ip_upgrade_results

get_lib_cells

get_macros

get_methodology_violations

get_nodes

get_parts

get_pins

get_pkgpin_nibbles

get_primitives

get_qor_suggestions

get_runs

get_site_pips

get_speed_models

get_timing_paths

list_hw_samples

remove_drc_checks

report_waivers

reset_property

scan_dr_hw_jtag

validate_board_files

Send Feedback

Partition:

create_partition_def   create_pr_configuration   create_reconfig_module
current_pr_configuration   delete_partition_defs   delete_pr_configurations
delete_reconfig_modules   get_partition_defs   get_pr_configurations
get_reconfig_modules   setup_pr_configurations

PinPlanning:

create_interface   create_port   delete_interface
make_diff_pair_ports   place_ports   remove_port
resize_port_bus   set_package_pin_val   split_diff_pair_ports

Platform:

combine_hw_platforms   open_hw_platform   validate_hw_platform
write_hw_platform   write_hw_platform_metadata

Power:

delete_power_results   power_opt_design   read_saif
report_power   report_power_opt   reset_operating_conditions
reset_switching_activity   set_operating_conditions   set_power_opt
set_switching_activity

Project:

add_files   add_peripheral_interface   apply_board_connection
archive_project   auto_detect_xpm   can_resolve_reference
check_syntax   close_design   close_project
compile_c   copy_constraints   copy_ip
create_dashboard_gadget   create_fileset   create_ip_run
create_peripheral   create_project   create_run
create_single_pass_run   create_xps   current_board_part
current_fileset   current_project   current_run
delete_dashboard_gadgets   delete_fileset   delete_ip_run
delete_runs   find_top   generate_peripheral
generate_target   get_board_parts   get_boards
get_dashboard_gadgets   get_files   get_filesets
get_ip_upgrade_results   get_ips   get_projects
get_runs   help   import_files
import_ip   import_synplify   import_xise
import_xst   launch_runs   list_targets
lock_design   make_wrapper   move_dashboard_gadget
move_files   open_checkpoint   open_example_project

open_io_design   open_project   open_run
refresh_design   refresh_meminit   reimport_files
remove_files   reorder_files   report_compile_order
reset_project   reset_runs   reset_target
save_constraints   save_constraints_as   save_project_as
set_part   set_speed_grade   synth_ip
update_compile_order   update_design   update_files
update_sw_parameters   upgrade_project   validate_board_files
wait_on_run   write_hwdef   write_ip_tcl
write_peripheral

projutils:

convert_ngc   copy_run   export_bd_synth
write_project_tcl

PropertyAndParameter:

create_property   filter   get_param
get_property   list_param   list_property
list_property_value   report_param   report_property
reset_param   reset_property   set_param
set_part   set_property

Report:

calc_config_time   check_timing   create_drc_violation
create_report_config   create_slack_histogram   delete_clock_networks_results
delete_report_configs   delete_timing_results   delete_utilization_results
generate_reports   get_msg_config   get_pplocs
get_report_configs   open_report   report_bus_skew
report_carry_chains   report_cdc   report_clock_interaction
report_clock_networks   report_clock_utilization   report_clocks
report_config_implementation report_config_timing   report_constant_paths
report_control_sets   report_datasheet   report_debug_core
report_design_analysis   report_disable_timing   report_drc
report_environment   report_exceptions   report_high_fanout_nets
report_hw_ddrmc   report_hw_mig   report_hw_softmc
report_incremental_reuse   report_io   report_methodology
report_operating_conditions report_param   report_phys_opt
report_power   report_pr_configuration_analysis report_property
report_pulse_width   report_qor_assessment   report_qor_suggestions
report_ram_utilization   report_route_status   report_sim_device
report_sim_version   report_ssn   report_switching_activity
report_synchronizer_mtbf   report_timing   report_timing_summary

Send Feedback   www.xilinx.com

report_transformed_primitives

report_utilization

report_waivers

reset_drc

reset_methodology

reset_msg_config

reset_msg_count

reset_ssn

reset_timing

set_msg_config

version

SDC:

all_clocks

all_inputs

all_outputs

all_registers

create_clock

create_generated_clock

current_design

current_instance

get_cells

get_clocks

get_hierarchy_separator

get_nets

get_pins

get_ports

group_path

set_case_analysis

set_clock_groups

set_clock_latency

set_clock_sense

set_clock_uncertainty

set_data_check

set_disable_timing

set_false_path

set_hierarchy_separator

set_input_delay

set_load

set_logic_dc

set_logic_one

set_logic_zero

set_max_delay

set_max_time_borrow

set_min_delay

set_multicycle_path

set_operating_conditions

set_output_delay

set_propagated_clock

set_units

Simulation:

add_bp

add_condition

add_files

add_force

checkpoint_vcd

close_saif

close_sim

close_vcd

compile_simlib

config_compile_simlib

create_fileset

current_frame

current_scope

current_sim

current_time

current_vcd

delete_fileset

describe

export_ip_user_files

export_simulation

flush_vcd

generate_hier_access

generate_mem_files

get_objects

get_scopes

get_simulators

get_stacks

get_value

import_files

launch_simulation

limit_vcd

log_saif

log_vcd

log_wave

ltrace

move_files

open_saif

open_vcd

open_wave_database

ptrace

read_saif

relaunch_sim

remove_bps

remove_conditions

remove_files

remove_forces

remove_conditions

report_conditions

report_drivers

report_bps

report_objects

report_scopes

report_frames

report_stacks

report_values

report_simlib_info

restart

run

reset_simulation

setup_ip_static_library

start_vcd

set_value

stop

stop_vcd

step

write_verilog

write_sdf

write_vhdl   write_xsim_coverage   xsim

SysGen:

create_sysgen   make_wrapper

Timing:

check_timing   config_design_analysis   config_timing_analysis
config_timing_corners   create_slack_histogram   delete_qor_suggestions
delete_timing_results   get_constant_paths   get_net_delays
get_qor_suggestions   get_timing_arcs   get_timing_paths
read_qor_suggestions   report_bus_skew   report_cdc
report_clock_interaction   report_clock_networks   report_clock_utilization
report_clocks   report_config_timing   report_constant_paths
report_datasheet   report_design_analysis   report_disable_timing
report_drc   report_exceptions   report_high_fanout_nets
report_methodology   report_pulse_width   report_qor_assessment
report_qor_suggestions   report_synchronizer_mtbf   report_timing
report_timing_summary   reset_timing   set_delay_model
set_disable_timing   set_external_delay   update_timing
write_inferred_xdc   write_qor_suggestions   write_sdf
write_xdc

ToolLaunch:

get_simulators   launch_chipscope_analyzer   launch_impact
launch_simulation

Tools:

iphys_opt_design   link_design   list_features
load_features   opt_design   phys_opt_design
place_design   read_iphys_opt_tcl   register_proc
report_pipeline_analysis   route_design   synth_design
unregister_proc   update_clock_routing   update_noc_qos
write_iphys_opt_tcl

Vitis:

combine_hw_platforms   open_hw_platform   validate_hw_platform
write_hw_platform   write_hw_platform_metadata

vivado_preferences:

current_vivado_preferences

Waiver:

create_waiver   delete_waivers   get_waivers
report_waivers   write_waivers

Waveform:

add_wave   add_wave_divider   add_wave_group
add_wave_marker   add_wave_virtual_bus   close_wave_config
create_wave_config   current_wave_config   get_wave_configs
get_waves   move_wave   open_wave_config
remove_wave   save_wave_config   select_wave_objects

XDC:

add_cells_to_pblock   add_to_power_rail   all_clocks
all_cpus   all_dsps   all_fanin
all_fanout   all_ffs   all_hsios
all_inputs   all_latches   all_outputs
all_rams   all_registers   connect_debug_cores
connect_debug_port   create_clock   create_debug_core
create_debug_port   create_generated_clock   create_macro
create_pblock   create_power_rail   create_property
create_waiver   current_design   current_instance
delete_macros   delete_pblocks   delete_power_rails
filter   get_bel_pins   get_bels
get_cells   get_clocks   get_debug_cores
get_debug_ports   get_generated_clocks   get_hierarchy_separator
get_iobanks   get_macros   get_nets
get_nodes   get_package_pins   get_path_groups
get_pblocks   get_pins   get_pips
get_pkgpin_bytegroups   get_pkgpin_nibbles   get_ports
get_power_rails   get_property   get_site_pins
get_site_pips   get_sites   get_slrs
get_speed_models   get_tiles   get_timing_arcs
get_wires   group_path   make_diff_pair_ports
remove_cells_from_pblock   remove_from_power_rail   reset_operating_conditions
reset_switching_activity   resize_pblock   set_bus_skew
set_case_analysis   set_clock_groups   set_clock_latency
set_clock_sense   set_clock_uncertainty   set_data_check
set_disable_timing   set_external_delay   set_false_path
set_hierarchy_separator   set_input_delay   set_input_jitter
set_load   set_logic_dc   set_logic_one
set_logic_unconnected   set_logic_zero   set_max_delay
set_max_time_borrow   set_min_delay   set_multicycle_path

Send Feedback

set_operating_conditions   set_output_delay   set_package_pin_val

set_power_opt   set_propagated_clock   set_property

set_switching_activity   set_system_jitter   set_units

update_macro

xilinxtclstore:

convert_ngc   copy_run   export_bd_synth

export_ip_user_files   export_simulation   generate_hier_access

setup_ip_static_library   write_project_tcl

Tcl Commands Listed Alphabetically

This chapter contains all SDC and Tcl commands, arranged alphabetically.

add_bp

Add breakpoint at a line of a HDL source

Syntax

```
add_bp [-quiet] [-verbose] <file_name> <line_number>
```

Returns

Return a new breakpoint object if there is not already a breakpoint set at the specified file line else returns the existing breakpoint object.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<file_name>	Filename to add the breakpoint
<line_number>	Line number of the given file to set the breakpoint

Categories

Simulation

Description

The `add_bp` command lets you add breakpoints to an HDL source file to pause the current simulation.

A breakpoint is a user-determined stopping point in the source code used for debugging the design. When simulating a design with breakpoints, simulation of the design stops at each breakpoint to let you examine values and verify the design behavior.

You can report breakpoints in the current simulation using the `report_bps` command, and remove existing breakpoints using the `remove_bps` command.

This command returns a new breakpoint object if there is not already a breakpoint set at the specified file line, or returns an existing breakpoint object if there is already a breakpoint defined for the specified file and line number.

> 💡 **TIP:** *You can capture the returned breakpoint object in a Tcl variable if needed.*

The `add_bp` command returns an error if the command fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file_name>` - (Required) The name of the HDL source file to add a breakpoint to.

`<line_number>` - (Required) The line number of the specified `<file_name>` to add the breakpoint to.

Examples

The following example adds a the breakpoint to the HDL source file at the specified line number:

```
add_bp C:/Data/ug937/sources/sinegen.vhd 137
```

See Also

- remove_bps
- report_bps

add_cells_to_pblock

Add cells to a Pblock

Syntax

```
add_cells_to_pblock [-top] [-add_primitives] [-clear_locs] [-quiet]
      [-verbose] <pblock> [<cells>...]
```

Usage

Name	Description
[-top]	Add the top level instance; This option can't be used with -cells, or -add_primitives options. You must specify either -cells or -top option.
[-add_primitives]	Assign to the pblock only primitive cells from the specified list of cells.
[-clear_locs]	Clear instance location constraints
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<pblock>	Pblock to add cells to
[<cells>]	Cells to add. You can't use this option with -top option. You must specify either -cells or -top option.

Categories

[Floorplan](#), [XDC](#)

Description

Adds specified logic instances to a Pblock in an open implemented design. Once cells have been added to a Pblock, you can place the Pblocks onto the fabric of the FPGA using the `resize_pblock` command. The `resize_pblock` command can also be used to manually move and resize Pblocks.

You can remove instances from the Pblock using the `remove_cells_from_pblock` command.

Arguments

`-top` - (Optional) Add the top level instance to create a Pblock for the whole design. You must either specify `<cells>` or the `-top` option to add objects to the Pblock.

`-add_primitives` - (Optional) Assign all primitives of the specified instances to a Pblock. This lets you select all cells in a hierarchical module, and assign only the leaf-cells of the selected cells to the specified Pblock.

Note: This option cannot be used with `-top`.

`-clear_locs` - (Optional) Clear instance location constraints for any cells that are already placed. This allows you to reset the LOC constraint for cells when defining new Pblocks for floorplanning purposes. When this option is not specified, any instances with assigned placement will not be unplaced as they are added to the Pblock.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<pblock>` - The name assigned to the Pblock.

`<cells>` - One or more cell objects to add to the specified Pblock.

Note: If `-top` is specified, you cannot also specify `<cells>`.

Examples

The following example creates a Pblock called pb_cpuEngine, and then adds only the leaf-cells found in the cpuEngine module, clearing placement constraints for placed instances:

```
create_pblock pb_cpuEngine
add_cells_to_pblock pb_cpuEngine [get_cells cpuEngine/*] \
    -add_primitives -clear_locs
```

See Also

- get_pblocks
- remove_cells_from_pblock
- resize_pblock

Send Feedback

add_condition

Conditionally execute Tcl commands

Syntax

```
add_condition [-name <arg>] [-radix <arg>] [-notrace] [-quiet] [-verbose]
    <condition_expression> <commands>
```

Returns

The condition object created.

Usage

Name	Description
[-name]	Assign a unique name (label) to a condition. Multiple conditions cannot be assigned the same name. If no name is specified, then a default label named as condition<id> is automatically created
[-radix]	Specifies which radix to use. Allowed values are: default, dec, bin, oct, hex, unsigned, ascii, smag.
[-notrace]	Turn off the logging of condition commands
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<condition_expression>	The condition expression when true executes the given commands
<commands>	Commands to execute upon condition

Categories

Simulation

Description

Add a condition that is evaluated by a specified condition, `<condition_expression>`, and runs a series of simulation Tcl commands when the condition is TRUE.

Conditions can be defined prior to starting the simulation. When a condition is added, the simulator evaluates the condition expression anytime a signal change is detected. When a specified condition expression becomes TRUE, the condition commands are run.

The `add_condition` command returns a condition identifier for the added condition, or returns an error if the command fails.

Send Feedback

Arguments

`-name <arg>` - (Optional) Provide a unique name for the condition. If no name is specified, then a default named is automatically created.

`-radix <arg>` - (Optional) Specifies the radix to use for the value of the condition. Allowed values are: `default`, `dec`, `bin`, `oct`, `hex`, `unsigned`, `ascii`, or `smag` (signed-magnitude).

Note: The radix `dec` indicates a signed decimal. Specify the radix `unsigned` when dealing with unsigned data.

`-notrace` - (Optional) Disable the logging of condition <commands> that are run when the condition is TRUE.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<condition_expression>` - (Required) Specify an expression for the condition. If the condition evaluates to true, the simulation will run the specified `<commands>`. Specific operators that can be used in condition expressions are "equal" (`==`), and "not-equal" (`!=`). Numeric comparison operators `<`, `>`, `<=`, and `>=` can also be used. Multiple filter expressions can be joined by AND and OR (`&&` and `||`).

`<commands>` - (Required) Specify the Tcl commands or Tcl procedure to run when the `<condition_expression>` is true. This command is surrounded by {} (braces). The command can include standard Tcl commands and simulation Tcl commands, except `run`, `restart`, and `step`. Tcl variables used in the condition expression are surrounded by quotes "" instead of {} so variable substitution can be performed. Refer to the *Vivado Design Suite User Guide: Using Tcl Scripting (UG894)* for more information on variable substitution.

Examples

The following example defines a condition named resetLow, that becomes true when the reset signal is low, and then puts a message to the standard output, and stops the current simulation:

```
add_condition -name resetLow {/testbench/reset == 0 } {
puts "Condition Reset was encountered at [current_time]. Stopping
simulation."
stop }
```

This next example defines a Tcl procedure, called myProc, that uses the `add_force` command to define `clk` and `reset` signal values, and print a standard message when it completes. A condition is then added that calls myProc when reset is low:

```
proc myProc {} {
  add_force clk {0 1} { 1 2} -repeat_every 4 -cancel_after 500
  add_force reset 1
  run 10 ns
  remove_force force2
  puts "Reached end of myProc"
}
add_condition -radix unsigned /top/reset==0 myproc
```

See Also

- add_force
- remove_conditions
- report_conditions
- stop

add_drc_checks

Add DRC rule check objects to a rule deck

Syntax

```
add_drc_checks [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
    -ruledeck <arg> [-quiet] [-verbose] [<patterns>]
```

Returns

Drc_check

Usage

Name	Description
[-of_objects]	Get 'rule_check' objects of these types: 'drc_ruledeck'.
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
-ruledeck	DRC rule deck to modify
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match the 'rule_check' objects against patterns. Default: *

Categories

DRC, Object

Description

Add design rule checks to the specified drc_ruledeck object.

A rule deck is a collection of design rule checks grouped for convenience, to be run with the `report_drc` command at different stages of the Xilinx design flow, such as during I/O planning or placement. The tool comes with a set of factory defined rule decks, but you can also create new user-defined rule decks with the `create_drc_ruledeck` command.

Use the `get_drc_ruledecks` command to return a list of the currently defined rule decks available for use in the `report_drc` command.

You can add standard factory defined rule checks to the rule deck, or add user-defined rule checks that were created using the `create_drc_check` command. Use the `get_drc_checks` command to get a list of checks that can be added to a rule deck.

Send Feedback

Checks can also be removed from a rule deck using the `remove_drc_checks` command.

Note: To temporarily disable a specific DRC rule, use the `set_property` command to set the `IS_ENABLED` property for the rule to false. This will disable the rule from being run in `report_drc`, without having to remove the rule from the rule deck. Use `reset_drc_check` to restore the rule to its default setting.

This command returns the list of design rule checks that were added to the rule deck.

Arguments

`-of_objects <arg>` - (Optional) Add the rule checks of the specified drc_ruledeck object to the specified rule deck. This has the effect of copying the rules from one rule deck into another.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter <args>` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by the search pattern, based on specified property values. You can find the properties on an object with the `report_property` or `list_property` commands.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-ruledeck <arg>` - (Required) The name of the rule deck to add the specified design rule checks to.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Add the design rule checks that match the specified patterns to the rule deck. The default pattern is the wildcard '*' which adds all rule checks to the specified rule deck. More than one pattern can be specified to find multiple rule checks based on different search criteria.

Note: You must enclose multiple search patterns in braces, {}, or quotes, "", to present the list as a single element.

Examples

The following example adds the rule checks matching the specified search pattern to the project_rules rule deck:

```
add_drc_checks -ruledeck project_rules {*DCI* *BUF*}
```

The following example creates a new rule deck called placer+, copies all of the rule checks from the placer_checks rule deck into the placer+ rule deck, then adds some additional checks:

```
create_drc_ruledeck placer+
add_drc_checks -of_objects [get_drc_ruledecks placer_checks] \
    -ruledeck placer+
add_drc_checks -ruledeck placer+ *IO*
```

The following example adds only the rule checks with a severity of Warning to the rule deck:

```
add_drc_checks -filter {SEVERITY == Warning} -ruledeck warn_only
```

See Also

- create_drc_check
- create_drc_ruledeck
- get_drc_checks
- get_drc_ruledecks
- list_property
- remove_drc_checks
- report_drc
- report_property
- reset_drc_check
- set_property

add_files

Add sources to the active fileset

Syntax

```
add_files [-fileset <arg>] [-of_objects <args>] [-norecurse]
    [-copy_to <arg>] [-force] [-scan_for_includes] [-quiet] [-verbose]
    [<files>...]
```

Returns

List of file objects that were added.

Usage

Name	Description
[-fileset]	Fileset name
[-of_objects]	Filesets or sub-designs or RMs to add the files to
[-norecurse]	Do not recursively search in specified directories
[-copy_to]	Copy the file to the specified directory before adding it to project
[-force]	Overwrite the existing file when -copy_to is used
[-scan_for_includes]	Scan and add any included files found in the fileset's RTL sources
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<files>]	Name of the files and/or directories to add. Must be specified if -scan_for_includes is not used.

Categories

Project, Simulation

Description

Adds one or more source files, or the source file contents of one or more directories, to the specified fileset in the current project. Valid source files include HDL sources (VHDL, Verilog, SystemVerilog, and related header files), netlist sources (DCP, EDIF, and NGC), and memory interface files (BMM, MIF, MEM, ELF).

IP and Block Design sources are not added through the `add_files` command. These are compound files that are supported by separate commands such as `import_ip`, `read_bd`, and `read_ip`.

Send Feedback

For every file added to a project the Vivado Design Suite attempts to store and maintain both a relative path and an absolute path to the file or directory. When a project is opened, these paths are used to locate the files and directories. By default the Vivado Design Suite applies a Relative First approach to resolving paths, searching the relative path first, then the absolute path. You can use the PATH_MODE property to change how the Vivado tool resolves file paths or properties for specific objects. For more information, see the *Vivado Design Suite Properties Reference Guide (UG912)*.

> **IMPORTANT!** *Adding multiple files one at a time can cause noticeable performance degradation. It is more efficient to use a single* `add_files` *command to import a list of files:*

```
add_files {file1 file2 file3 ... fileN}
```

The Vivado tool does not read the contents of a file automatically when the file is added to the project with `add_files`, but rather reads the file contents when they are needed. For instance, a constraints file is not read when added to the design until needed by synthesis, timing, or implementation. To read the file at the time of adding it to the design, use the `read_xxx` command instead.

> **TIP:** *When running the Vivado tool in Non-Project mode, in which there is no project file to maintain and manage the various project source files, you should use the* `read_xxx` *commands to read the contents of source files into the in-memory design. Refer to the Vivado Design Suite User Guide: Design Flows Overview (UG892) for more information on Non-Project mode.*

The `add_files` command adds them by reference to the specified fileset. This is different from the `import_files` command, which copies the file into the local project folders as well as adding them to the specified fileset.

This command returns the files that were added, or returns an error if it fails.

Arguments

`-fileset` <name> - (Optional) The fileset to which the specified source files should be added. An error is returned if the specified fileset does not exist. If no fileset is specified the files are added to the source fileset by default.

`-norecurse` - (Optional) Do not recurse through subdirectories of any specified directories. Without this argument, the tool searches through any subdirectories for additional source files that can be added to a project.

`-copy_to` <arg> - (Optional) Copy the selected files to the specified directory before adding the files to the project. This option lets you move files from their current location to a new folder to reference as part of the source structure for a project.

`-force` - (optional) Overwrite any existing files of the same name as the specified files, in the destination directory, when you use the `-copy_to` option.

`-scan_for_includes` - (Optional) Scan Verilog source files for any `'include` statements and add these referenced files to the specified fileset. By default, `'include` files are not added to the fileset.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<files>` - (Optional) One or more file names or directory names to be added to the fileset. If a directory name is specified, all valid source files found in the directory, and in subdirectories of the directory, are added to the fileset.

Note: If the path is not specified as part of the file name, the tool will search for the specified file in the current working directory and then in the directory from which the tool was launched.

Examples

The following example adds a file called rtl.v to the current project:

```
add_files rtl.v
```

In the preceding example the tool looks for the rtl.v file in the current working directory since no file path is specified, and the file is added to the source fileset as a default since no fileset is specified.

The following example adds a file called `top.xdc` to the `constrs_1` constraint fileset, as well as any appropriate source files found in the project_1 directory, and its subdirectories:

```
add_files -fileset constrs_1 -quiet c:/Design/top.xdc c:/Design/project_1
```

In addition, the tool ignores any command line errors because the `-quiet` argument was specified.

If the `-norecurse` option had been specified then only constraint files found in the `project_1` directory would have been added, but subdirectories would not be searched.

The following example adds an existing IP core file to the current project:

```
add_files -norecurse C:/Data/ip/c_addsub_v11_0_0.xci
```

Note: Use the `import_ip` command to import the IP file into the local project folders.

Send Feedback

The following example reads a top-level design netlist, and the char_fifo IP in a Non-Project Mode design:

```
# Read top-level EDIF and IP DCP
read_edif ./sources/wave_gen.edif
add_files ./my_IP/char_fifo/char_fifo.xci
```

Note: Either `add_files` or `read_ip` can be used reading in an IP core. All output products of the IP, including a design checkpoint (DCP), will be read as needed.

The following example adds an existing DSP module, created in System Generator, into the current project:

```
add_files C:/Data/model1.mdl
```

Note: Use the `create_sysgen` command to use System Generator to create a new DSP module.

See Also

- create_sysgen
- import_files
- import_ip
- read_ip
- read_verilog
- read_vhdl
- read_xdc

add_force

Force value of signal, wire, or reg to a specified value

Syntax

```
add_force [-radix <arg>] [-repeat_every <arg>] [-cancel_after <arg>]
     [-quiet] [-verbose] <hdl_object> <values>...
```

Returns

The force objects added.

Usage

Name	Description
`[-radix]`	Specifies which radix to use. Allowed values are: default, dec, bin, oct, hex, unsigned, ascii, smag
`[-repeat_every]`	Repeat every time duration
`[-cancel_after]`	Cancel after time offset
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<hdl_object>`	Specifies the object upon which to add a force
`<values>`	Adds a value and time offset to the force: {value [time_offset] }

Categories

Simulation

Description

Force the value of a signal, wire, or reg to a certain value during simulation.

The `add_force` command has the same effect as the Verilog `force/release` commands in the test bench or the module definition. It forces an HDL object to hold the specified value for the specified time, or until released by the `-cancel_after` option, or the `remove_forces` command.

> ⭐ **IMPORTANT!** *If there are Verilog `force/release` statements on an HDL object in the test bench or module, these commands are overridden by the Tcl `add_force` command. When the Tcl force expires or is released, the HDL object resumes normal operation in the simulation, including the application of any Verilog forces.*

Send Feedback

This command returns the name of the force object created, or returns an error if the command failed. The name of the returned force object is important when using the `remove_forces` command, and should be captured in a Tcl variable for later recall, as shown in the examples.

Arguments

`-radix <arg>` - (Optional) The radix used when specifying the `<values>`. Allowed values are: `default`, `dec`, `bin`, `oct`, `hex`, `unsigned`, `ascii`, or `smag`. The default radix is binary, unless the specified HDL object type has an overriding radix defined.

`-repeat_every <arg>` - (Optional) Causes the `add_force` to repeat over some specified increment of time. This can be used to create a recurring force on the specified `<hdl_object>`.

Note: The specified time must be greater than the time period defined by any `{<value> <time>}` pairs defined by `<values>`, or an error will be returned.

`-cancel_after <arg>` - (Optional) Cancels the force effect after the specified period of time from the `current_time`. This has the same effect as applying the `remove_forces` command after the specified period of time.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<hdl_object>` - (Required) Specifies a single HDL object to force the value of. The object can be specified by name, or can be returned as an object by the `get_objects` command. Valid objects are signal, wire, and reg.

`<values>` - (Required) The value to force the HDL object to. A single value can be specified, and the value will be held during simulation until the force is removed either through the use of the `-cancel_after` option, or through the use of the `remove_forces` command.

The specified `<value>` depends on the type of the `<hdl_object>`. HDL object types include: "logic", floating point, VHDL enumerated, and VHDL integral. For all but "logic" the `-radix` option is ignored.

- "Logic" does not refer to an actual HDL object type, but means any object whose values are similar to those of VHDL std_logic, such as:
  - the Verilog implicit 4-state bit type,
  - the VHDL bit and std_logic predefined types,

- any VHDL enumeration type which is a subset of std_logic, including the character literals '0' and '1'.

- For logic types the value depends on the radix:

  - If the specified value has fewer bits than the logic type expects, the value is zero extended, but not sign extended, to match the expected length.

  - If the specified value has more bits than the logic type expects, the extra bits on the MSB side should all be zeros, or the Vivado simulator will return a "size mismatch" error.

- Accepted values for floating point objects are floating point values.

- The accepted value for non-logic VHDL enumerated types is a scalar value from the enumerated set of values, without single quotes in the case of characters.

- Accepted values for VHDL integral types is a signed decimal integer in the range accepted by the type.

The `<value>` can also be specified as `{<value> <time>}` pairs, which forces the HDL object to hold a specified `<value>` for a specified period of `<time>` from the current time, then hold the next `<value>` for the next period of `<time>`, until the end of the `{<value> <time>}` pairs.

Note: In `{<value> <time>}` pairs, the `<time>` is optional in the first pair, and will be assumed to be 0 if it is not specified. In all subsequent `{<value> <time>}` pairs, the `<time>` is required.

The `<time>` specified in `{<value> <time>}` pairs is defined relative to the current simulation time, and indicates a period of time from the `current_time`. For example, if the current simulation time is 1000 ns, a `<time>` of 20 ns defines a period from the current time to 1020 ns.

> ⭐ **IMPORTANT!** *The `<times>` must be specified in increasing order on the simulation time line, and may not be repeated, or an error will occur.*

The `<time>` is specified in the default TIME_UNIT of the current simulation, or can be specified with the time unit included, with no white space. Valid units of time are: fs, ps, ns, us, ms, or s. A time of 50 defines a period of 50 ns (the default). A `<time>` of `50ps` defines a period of 50 picoseconds.

Examples

The following example forces the reset signal high at 300 nanoseconds, using the default radix, and captures the name of the returned force object in a Tcl variable which can be used to later remove the force:

```
set for10 [ add_force reset 1 300 ]
```

The following example shows the use of `{<value> <time>}` pairs, repeated periodically, and canceled after a specified time.

```
add_force mySig {0} {1 50} {0 100} {1 150} -repeat_every 200
    -cancel_after 10000
```

Note: In the preceding example, the first `{<value> <time>}` pair does not include a time. This indicates that the specified value, 0, is applied at time 0 (the `current_time`).

See Also

- current_time
- get_objects
- remove_forces

add_hw_hbm_pc

Select Pseudo Channel to add to Activity Monitor on the specified hardware HBM(s). Must specify the Memory Controller number first, follow by Pseudo Channel number.

Syntax

```
add_hw_hbm_pc [-quiet] [-verbose] <mc_num> <pc_num> <hw_objects>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<mc_num>	Memory Controller number to select: 0 to 7
<pc_num>	Pseudo Channel number to select: 0 or 1
<hw_objects>	hardware objects

Categories

[Hardware](#)

Description

The HBM activity monitor (amon) can be used to gain real-time access to performance monitoring and temperature sensors certain Xilinx® UltraScale+ devices that include an integrated High-Bandwidth Memory (HBM) controller. Refer to the LogiCORE IP Product Guide: AXI High Bandwidth Memory Controller ([PG276](#)) for more information on this core. The HBM controller and memory stacks contain both performance counters and temperature sensors that can be accessed through the HBM activity monitor from within the Xilinx Vivado hardware manager. Each HBM stack is split into eight independent memory channels, each of which is further divided into two 64-bit pseudo channels (pc).

After configuring an HBM enabled device with a design that contains instances of the AXI High Bandwidth Memory Controller, the HBM cores will be visible in the Vivado hardware manager. The `add_hw_hbm_pc` command lets you specify a memory channel (mc)/pseudo channel (pc) to monitor in the HBM activity monitor prior to using the `run_hw_hbm_amon` command.

> 💡 **TIP:** *The HBM activity monitor must not be running when adding or removing pseudo channels.*

This command returns nothing if successful, or returns an error if it fails.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the set_msg_config command.

<mc_num> - (Required) Specifies one of the 8 memory channels of the HBM controller, numbered 0 to 7, to add to the HBM activity monitor.

<pc_num> - (Required) Specifies one of the 2 pseudo channels, numbered 0 or 1.

<hw_objects> - (Required) Specifies the HBM controller (hw_hbms) to monitor.

Examples

The following example adds the specified memory channels/pseudo channels on the defined HBM controller (hw_hbm) object, and then runs the HBM activity monitor in the Vivado hardware manager:

```
set hbm_mon [get_hw_hbms *HBM_2]
add_hw_hbm_pc 0 0 $hbm_mon
add_hw_hbm_pc 0 1 $hbm_mon
add_hw_hbm_pc 1 0 $hbm_mon
add_hw_hbm_pc 1 1 $hbm_mon
add_hw_hbm_pc 2 0 $hbm_mon
add_hw_hbm_pc 2 1 $hbm_mon
add_hw_hbm_pc 3 0 $hbm_mon
add_hw_hbm_pc 3 1 $hbm_mon
run_hw_hbm_amon $hbm_mon
```

See Also

- commit_hw_hbm
- current_hw_device
- current_hw_target
- get_hw_hbms
- pause_hw_hbm_amon
- refresh_hw_hbm
- remove_hw_hbm_pc
- resume_hw_hbm_amon
- run_hw_hbm_amon
- stop_hw_hbm_amon

Send Feedback

add_hw_probe_enum

Add an enumerated name-value pair to a hw_probe enumeration.

Syntax

```
add_hw_probe_enum [-no_gui_update] [-dict <args>] [-quiet] [-verbose]
     <name> <value> <hw_probe>
```

Usage

Name	Description
[-no_gui_update]	Defer GUI update.
[-dict]	List of parameter name-value pairs.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<name>	Enumerated name.
<value>	Explicit value.
<hw_probe>	hw_probe object.

Categories

Hardware

Description

Assign enumerated name/value pairs to specified hardware probe objects.

This command is intended to make it easier to monitor the states of signals in the Vivado logic analyzer. The command lets you define a set of states, or enumerated names to be associated with specific values that may be found on a hw_probe object. This lets you monitor state machine probes and some other types of probes, by comparing symbolic names with trigger values and waveform data values.

The enumerated name is added as an ENUM.NAME property on the specified hw_probe object, and associated with the specified bit value on the probe. Enumerated names can be used to specify trigger/capture compare values for hw_probes.

> **TIP:** *Enumerated names are displayed in the waveform viewer of the Vivado logic analyzer. Display of the enumerated names can be disabled on a per probe basis. Refer to the Vivado Design Suite User Guide: Programming and Debugging (UG908) for more information on the waveform viewer.*

This command returns the enumerated name property, or returns an error if it fails.

Send Feedback

Arguments

`-no_gui_update` - (Optional) Do not update the GUI in the Vivado logic analyzer to reflect the enumerated values of the probe.

`-dict` - (Optional) Use this option to specify a dictionary of enumerated `<name>` `<value>` pairs on a `hw_probe`. Multiple `<name>` `<value>` pairs must be enclosed in quotes, `""`, or braces, `{}`.

```
-dict "name1 value1 name2 value2 ... nameN valueN"
```

> **TIP:** *Use the* `-dict` *option in place of the* `<name>` *and* `<value>` *arguments when specifying multiple enumerated values.*

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - (Required) Specify the name for the ENUM property associated with the specified `<value>`, on the hw_probe object. The enumerated name property is case insensitive. The specified name will be used whenever the bit value on the hw_probe matches the specified `<value>`.

`<value>` - (Required) Specify the bit value on the hw_probe object to associate with the defined enumerated `<name>`. Values can be defined using binary, octal, hex, signed and unsigned values.

> **IMPORTANT!** *Binary bit-values 'x' and edge bit values (F,B,RT) can not be specified.*

`<hw_probe>` - (Required) Specify the hw_probe object to assign the enumerated name property to.

Examples

The following example uses the `-dict` option to define the enumerated name/value pairs for the specified hw_probe object:

```
add_hw_probe_enum -dict {ZERO eq5'h00 RED eq5'h12 GREEN eq5'h13 \
   BLUE eq5'h14 WHITE eq5'h15 YELLOW eq5'h16  GREY eq5'h17} \
   [get_hw_probes op1 -of_objects [current_hw_ila]]
```

The following example defines the enumerated name/value pairs for the specified hw_probe object:

```
add_hw_probe_enum  ZERO eq5'h00 [get_hw_probes op1 \
   -of_objects [current_hw_ila]]
add_hw_probe_enum  RED eq5'h12 [get_hw_probes op1 \
   -of_objects [current_hw_ila]]
add_hw_probe_enum  GREEN eq5'h13 [get_hw_probes op1 \
   -of_objects [current_hw_ila]]
add_hw_probe_enum  BLUE eq5'h14 [get_hw_probes op1 \
   -of_objects [current_hw_ila]]
add_hw_probe_enum  WHITE eq5'h15 [get_hw_probes op1 \
   -of_objects [current_hw_ila]]
add_hw_probe_enum  YELLOW eq5'h16 [get_hw_probes op1 \
   -of_objects [current_hw_ila]]
add_hw_probe_enum  GREY eq5'h17 [get_hw_probes op1 \
   -of_objects [current_hw_ila]]
```

The following example returns the ENUM property assigned to the specified hw_probe object:

```
report_property [get_hw_probes op1 -of_objects [current_hw_ila]] ENUM*
Property              Type      Read-only  Visible   Value
ENUM.ZERO             string    true       true      eq5'h00
ENUM.RED              string    true       true      eq5'h12
ENUM.GREEN            string    true       true      eq5'h13
ENUM.BLUE             string    true       true      eq5'h14
ENUM.WHITE            string    true       true      eq5'h15
ENUM.YELLOW           string    true       true      eq5'h16
ENUM.GREY             string    true       true      eq5'h17
```

See Also

- current_hw_device
- current_hw_ila
- get_hw_devices
- get_hw_ilas
- get_hw_probes
- get_hw_vios
- remove_hw_probe_enum
- report_property

Send Feedback

add_peripheral_interface

Add a new bus interface to a peripheral.

Syntax

```
add_peripheral_interface -interface_mode <arg> -axi_type <arg> [-quiet]
    [-verbose] <name> <peripheral>
```

Usage

Name	Description
`-interface_mode`	Mode of an interface, supported option - master,slave.
`-axi_type`	Type of a axi interface, supported option - lite,full,stream.
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<name>`	Name to initialize the newly added element e.g S1_AXI, M1_AXI
`<peripheral>`	Peripheral object

Categories

Project, IPFlow, CreatePeripheral

Description

Add an AXI bus interface to a peripheral created with the `create_peripheral` command.

Arguments

`-interface_mode [ master | slave ]` - (Optional) Specify the interface as a slave or master interface. The master interface generates out-bound AXI transactions and thus is the source of an AXI transfer. A slave interface receives in-bound AXI transactions and is the target of an AXI transfer.

`-axi_type <arg>` - (Optional) Type of AXI interface to add. The supported values are: full, lite, and stream.

- The full AXI4 interface is for memory mapped interfaces allowing bursts of up to 256 data transfer cycles with just a single address phase.

- The AXI4-Lite protocol is a subset of the AXI4 protocol intended for communication with simpler, smaller control register-style interfaces in components.

- The AXI4-Stream protocol is designed for unidirectional data transfers from master to slave with greatly reduced signal routing.

Note: For more information on AXI interfaces refer to the *AXI Reference Guide (UG761)*.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - (Required) User-specified name of the interface to add.

`<peripheral>` - (Required) The peripheral object to add the interface to. The peripheral is created with the `create_peripheral` command, which should be captured in a Tcl variable to facilitate further processing by this and other related commands. See the example below.

Example

This example creates a new AXI peripheral, with the VLNV attribute as specified, and captures the peripheral object in a Tcl variable for later processing, then adds AXI slave interfaces to the peripheral:

```
set perifObj [ create_peripheral {myCompany.com} {user} {testAXI1} {1.3} \
    -dir {C:/Data/new_periph} ]
add_peripheral_interface {S0_AXI} -interface_mode {slave} \
    -axi_type {lite} $perifObj
add_peripheral_interface {S1_AXI} -interface_mode {slave} \
    -axi_type {lite} $perifObj
add_peripheral_interface {S2_AXI} -interface_mode {slave} \
    -axi_type {lite} $perifObj
```

See Also

- create_peripheral
- generate_peripheral
- write_peripheral

add_to_power_rail

Add power sources to power Rail

Syntax

```
add_to_power_rail [-power_sources <args>] [-quiet] [-verbose] <power_rail>
```

Usage

Name	Description
[-power_sources]	List of power_sources to add. Can be power rails and/or power supplies
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<power_rail>	Power rail to add sources to

Categories

XDC, Floorplan

Send Feedback

add_wave

Add new waves

Syntax

```
add_wave [-into <args>] [-at_wave <args>] [-after_wave <args>]
    [-before_wave <args>] [-reverse] [-radix <arg>] [-color <arg>]
    [-name <arg>] [-recursive] [-r] [-regexp] [-nocase] [-quiet] [-verbose]
    <items>...
```

Returns

The new waves.

Usage

Name	Description
[-into]	the wave configuration, group, or virtual bus into which the new wave object(s) will be inserted.
[-at_wave]	inserts the new wave object(s) into the specified wave object, or after the specified wave object if not a group or virtual bus
[-after_wave]	inserts the new wave objects(s) after the specified wave object
[-before_wave]	inserts the new wave objects(s) before the specified wave object
[-reverse]	reverses the displayed bit order of the new wave objects(s)
[-radix]	sets the displayed radix of the new wave object(s) to the specified radix. Allowed values are: default, dec, bin, oct, hex, unsigned, ascii, smag
[-color]	sets the displayed color of the new wave object(s) to the specified color, which can be a standard color name or a string of the form #RRGGBB
[-name]	sets the displayed name of the single new wave object to the specified string
[-recursive]	if the design object is a scope, this option specifies that wave objects for all design objects under that scope should be created
[-r]	if the design object is a scope, this option specifies that wave objects for all design objects under that scope should be created
[-regexp]	interprets <items> using regular expressions
[-nocase]	only when regexp is used, performs a case insensitive match
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<items>	the design objects from which to create wave objects

Send Feedback

Categories

[Waveform](#)

Description

The `add_wave` command creates one or more new design-based wave objects.

This command returns the name of the newly-created wave object(s).

Note: This command can only be used when running a simulation. At a minimum, you must specify an `item`, which is an HDL object (signal) within the simulation project. In the Vivado interface, the object would display in the Objects Window.

Arguments

`-into <wcfgGroupVbusObj>` - (Optional) Specifies the wave configuration, group, or virtual bus into which the new wave object(s) are inserted. If `<wcfgGroupVbusObj>` is a string instead of an object, it is treated as the name of a group in the current WCFG. If no such group is found, the tool searches the names of the virtual buses of the current WCFG. If still not found, the tool searches the names of all WCFG objects. If no `-into` object is specified, the current wave configuration is assumed.

`-at_wave <waveObj>` - (Optional) Adds a wave object at a specified wave object. If `<waveObj>` is a string, it is treated as the display name of a wave object.

`-after_wave <waveObj>` - (Optional) Adds a wave object after a specified wave object. If `<waveObj>` is a string, it is treated as the display name of a wave object.

`-before_wave <waveObj>` - (Optional) Adds a wave object before a specified wave object. If `<waveObj>` is a string, it is treated as the display name of a wave object.

`-reverse` - (Optional) Sets the `IS_REVERSED` property of the new wave object(s) to `true`.

`-radix <arg>` - (Optional) Sets the radix property of the new wave object(s) to radix. Allowed values are: `default`, `dec`, `bin`, `oct`, `hex`, `unsigned`, `ascii`, or `smag`.

`-color <arg>` - (Optional) Sets the color property of the new wave object(s) to color, which is either a pre-defined color name or a color specified by a six-digit RGB format (#RRGGBB).

`-name <arg>` - (Optional) Sets the `DISPLAY_NAME` property of the new wave object to the specified name. It is an error for there to be more than one new wave object being created.

`-recursive | -r` - (Optional) If <items> specifies a scope, this option specifies that all sub-scopes of that scope should also be added.

`-regexp` - (Optional) Specifies that `<items>` are written as regular expressions Xilinx regular expression commands are always anchored to the start of the search string. You can add ".*" to the beginning of the search string to widen the search. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command regexp is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<items>` - (Required) Add waves for the specified HDL objects in the current simulation.

Examples

The following example defines a wave group, defines a color for that group, and adds all signals that begin with the letter 's', and has at least one character following it followed by 0 or more characters, to the wave window in the wave group:

```
set groupColor YELLOW
set AXIS_ID [add_wave_group "Streaming Data"]
add_wave -into $AXIS_ID -color $groupColor -regexp s.*
```

Add the dout_tvalid signal from the rsb_design_testbench to the existing simulation waveform configuration:

```
add_wave dout_tvalid /rsb_design_testbench/dout_tvalid
```

See Also

- add_wave_divider
- add_wave_group
- add_wave_marker
- add_wave_virtual_bus

add_wave_divider

Add a new divider

Syntax

```
add_wave_divider [-into <args>] [-at_wave <args>] [-after_wave <args>]
    [-before_wave <args>] [-color <arg>] [-quiet] [-verbose] [<name>]
```

Returns

The new divider.

Usage

Name	Description
[-into]	the wave configuration or group into which the new divider will be inserted.
[-at_wave]	inserts the new divider into the specified wave object, or after the specified wave object if not a group
[-after_wave]	inserts the new divider after the specified wave object
[-before_wave]	inserts the new divider before the specified wave object
[-color]	sets the displayed color of the new divider to the specified color, which can be a standard color name or a string of the form #RRGGBB Default: default
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<name>]	the displayed name of the new divider to the specified string Default: new_divider

Categories

Waveform

Description

Creates a wave divider in the wave form viewer. The wave divider can be used to separate groups of related objects, for easier viewing.

The wave divider can be added into a specified or current waveform configuration (WCFG) at the specified location. If no location is specified the wave divider is inserted at the end of the waveform configuration.

This command returns the name of the newly-created wave divider.

Note: This command can only be used when running a simulation.

Send Feedback

Arguments

`-into <wcfgGroupVbusObj>` - (Optional) Specifies the wave configuration, group, or virtual bus into which the new wave divider object(s) are inserted. If `<wcfgGroupVbusObj>` is a string instead of an object, it is treated as the name of a group in the current WCFG. If no such group is found, the tool searches the names of the virtual buses of the current WCFG. If still not found, the tool searches the names of all WCFG objects. If no `-into` object is specified, the current wave configuration is assumed.

`-at_wave <waveObj>` - (Optional) Adds a wave divider at a specified wave object. If <waveObj> is a string, it is treated as the display name of a wave object.

`-after_wave <waveObj>` - (Optional) Adds a wave divider after a specified wave object. If <waveObj> is a string, it is treated as the display name of a wave object.

`-before_wave <waveObj>` - (Optional) Adds a wave divider before a specified wave object. If <waveObj> is a string, it is treated as the display name of a wave object.

`-color <arg>` - (Optional) Sets the color property of the new wave object(s) to a pre-defined color name or a color specified by a six-digit RGB format (RRGGBB).

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - (Optional) Creates a divider with the specified display name. The default name is `new_divider`.

Examples

The following example inserts a wave divider named Div1, after the CLK wave object:

```
add_wave_divider -after_wave CLK Div1
```

See Also

- add_wave
- add_wave_group
- add_wave_marker
- add_wave_virtual_bus

add_wave_group

Add a new group

Syntax

```
add_wave_group [-into <args>] [-at_wave <args>] [-after_wave <args>]
    [-before_wave <args>] [-quiet] [-verbose] [<name>]
```

Returns

The new group.

Usage

Name	Description
[-into]	the wave configuration or group into which the new group will be inserted.
[-at_wave]	inserts the new group into the specified wave object, or after the specified wave object if not a group
[-after_wave]	inserts the new group after the specified wave object
[-before_wave]	inserts the new group before the specified wave object
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<name>]	the displayed name of the new group to the specified string Default: new_group

Categories

Waveform

Description

Creates a wave group into a specified or current waveform configuration. New wave objects and wave_dividers can be added into the wave group to build up the waveform display.

The wave group can be inserted at a specified location. If no location is specified the group is inserted at the end of the specified waveform configuration.

The command returns the name of the newly created wave group object.

Note: This command can only be used when running a simulation.

Send Feedback

Arguments

`-into <wcfgGroupVbusObj>` - (Optional) Specifies the wave configuration, group, or virtual bus into which the new wave group object(s) are inserted. If `<wcfgGroupVbusObj>` is a string instead of an object, it is treated as the name of a group in the current WCFG. If no such group is found, the tool searches the names of the virtual buses of the current WCFG. If still not found, the tool searches the names of all WCFG objects. If no `-into` object is specified, the current wave configuration is assumed.

`-at_wave <waveObj>` - (Optional) Adds a wave group at a specified wave object. If `<waveObj>` is a string, it is treated as the display name of a wave object.

`-after_wave <waveObj>` - (Optional) Adds a wave group after a specified wave object. If `<waveObj>` is a string, it is treated as the display name of a wave object.

`-before_wave <waveObj>` - (Optional) Adds a wave group before a specified wave object. If `<waveObj>` is a string, it is treated as the display name of a wave object.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - (Optional) Creates a wave group with the specified display name. The default name is `new_group`.

Examples

Add a clk to the existing waveform configuration:

```
add_wave_group clk
  group10
```

See Also

- [add_wave](#)
- [add_wave_divider](#)
- [add_wave_marker](#)
- [add_wave_virtual_bus](#)

add_wave_marker

Create a new wave marker

Syntax

```
add_wave_marker [-into <arg>] [-name <arg>] [-quiet] [-verbose] [<time>]
    [<unit>]
```

Returns

The new created marker.

Usage

Name	Description
[-into]	the wave configuration in which to create the marker
[-name]	sets the name of the new marker to the specified string
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<time>]	the numerical portion of the new marker's time Default: 0
[<unit>]	the time unit portion of the new marker's time. Allowed values are fs, ps, ns, us, ms, and s.

Categories

Waveform

Description

Creates a wave marker at the specified time and of the specified name in the current waveform configuration.

This command returns nothing.

Note: This command can only be used when running a simulation.

Arguments

`-into` <wcfg> - (Optional) Specifies the WCFG object into which the new wave marker is inserted. If `-into` is not specified, the wave marker is added to the current wave configuration.

`-name <arg>` - (Optional) Creates a marker with the specified name. The default name is `new_marker`.

<time> - (Optional) Is the simulation runtime within the waveform at which to set the marker. The default is time 0.

<unit> - (Optional) Is the time unit. Allowable units are `s`, `ms`, `us`, `ns`, and `ps`. The default is the time unit used in the specified waveform configuration.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

Add a marker to the existing waveform configuration at 500ns:

```
add_wave_marker 500 ns
```

See Also

- add_wave
- add_wave_divider
- add_wave_group
- add_wave_virtual_bus

add_wave_virtual_bus

Add a new virtual bus

Syntax

```
add_wave_virtual_bus [-into <args>] [-at_wave <args>] [-after_wave <args>]
    [-before_wave <args>] [-reverse] [-radix <arg>] [-color <arg>] [-quiet]
    [-verbose] [<name>]
```

Returns

The new virtual bus.

Usage

Name	Description
[-into]	the wave configuration, group, or virtual bus into which the new virtual bus will be inserted.
[-at_wave]	inserts the new virtual bus into the specified wave object, or after the specified wave object if not a group or virtual bus
[-after_wave]	inserts the new virtual bus after the specified wave object
[-before_wave]	inserts the new virtual bus before the specified wave object
[-reverse]	reverses the displayed bit order of the new virtual bus
[-radix]	sets the displayed radix of the new virtual bus to the specified radix. Allowed values are: default, dec, bin, oct, hex, unsigned, ascii, smag
[-color]	sets the displayed color of the new virtual bus to the specified color, which can be a standard color name or a string of the form #RRGGBB Default: default
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<name>]	the displayed name of the new virtual bus to the specified string Default: new_virtual_bus

Categories

Waveform

Description

The `add_wave_virtual_bus` command creates a new virtual bus of the specified `<name>`. The command inserts the virtual bus into the wave configuration (WCFG) where specified, or by default at the bottom of the existing WCFG. It returns a `vb###` for the newly-created virtual bus.

Note: This command can only be used when running a simulation. At a minimum, you must specify a name, which is the name of the new virtual bus

Arguments

`-into` <wcfgGroupVbusObj> - (Optional) Specifies the wave configuration, group, or virtual bus into which the new virtual bus object(s) are inserted. If <wcfgGroupVbusObj> is a string instead of an object, it is treated as the name of a group in the current WCFG. If no such group is found, the tool searches the names of the virtual buses of the current WCFG. If still not found, the tool searches the names of all WCFG objects. If no `-into` object is specified, the current wave configuration is assumed.

`-at_wave` <waveObj> - (Optional) Adds a virtual bus object at a specified wave object. If <waveObj> is a string, it is treated as the display name of a wave object.

`-after_wave` <waveObj> - (Optional) Adds a virtual bus object after a specified wave object. If <waveObj> is a string, it is treated as the display name of a wave object.

`-before_wave` <waveObj> - (Optional) Adds a virtual bus object before a specified wave object. If <waveObj> is a string, it is treated as the display name of a wave object.

`-reverse` - (Optional) Sets the `IS_REVERSED` property of the new virtual bus object(s) to `true`.

`-radix <arg>` - (Optional) Sets the radix property of the new virtual bus object(s) to radix. Allowed values are: `default`, `dec`, `bin`, `oct`, `hex`, `unsigned`, `ascii`, or `smag`.

`-color <arg>` - (Optional) Sets the color property of the new wave object(s) to the specified color, which can be a pre-defined color name or a color specified by a six-digit RGB format (RRGGBB).

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<name> - (Optional) Sets the `DISPLAY_NAME` property of the new virtual bus object to <name>.

Examples

Add a virtual bus of the name `dout_tvalid` to the end of the current waveform configuration:

```
add_wave_virtual_bus dout_tvalid
```

See Also

- add_wave_divider
- add_wave_group
- add_wave_marker
- add_wave

all_clocks

Get a list of all clocks in the current design

Syntax

```
all_clocks [-quiet] [-verbose]
```

Returns

List of clock objects.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[SDC](#), [XDC](#)

Description

Returns a list of all clocks that have been declared in the current design.

To get a list of specific clocks in the design, use the `get_clocks` command, or use the `filter` command to filter the results returned by `all_clocks`.

Clocks can be defined by using the `create_clock` or `create_generated_clock` commands.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example shows all clocks in the sample CPU netlist project:

```
% all_clocks
cpuClk wbClk usbClk phy_clk_pad_0_i phy_clk_pad_1_i fftClk
```

The following example applies the set_propagated_clock command to all clocks, and also demonstrates how the returned list (all_clocks) can be passed to another command:

```
% set_propagated_clock [all_clocks]
```

See Also

- create_clock
- create_generated_clock
- filter
- get_clocks
- set_propagated_clock

Send Feedback

all_cpus

Get a list of cpu cells in the current design

Syntax

```
all_cpus [-quiet] [-verbose]
```

Returns

List of cpu cell objects.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[XDC](XDC)

Description

Returns a list of all CPU cell objects in the current design. Creates a list of all the CPU cell objects that have been declared in the current design.

The list of CPUs returned by `all_cpus` can also be limited or reduced by the `filter` command to filter according to properties assigned to the CPU cell objects. Properties of an object can be returned by the `list_property` or `report_property` commands.

The `all_cpus` command is scoped to return the objects hierarchically, from the top-level of the design or from the level of the current instance. By default the current instance is defined as the top level of the design, but can be changed by using the `current_instance` command.

Note: This command returns a list of CPU cell objects

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

Send Feedback

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example returns all CPU objects in the current design:

```
all_cpus
```

The following example shows how the list returned can be passed to another command:

```
set_false_path -from [all_cpus] -to [all_registers]
```

See Also

- all_dsps
- all_hsios
- all_registers
- current_instance
- filter
- get_cells
- list_property
- report_property
- set_false_path

Send Feedback

all_dsps

Get a list of dsp cells in the current design

Syntax

```
all_dsps [-quiet] [-verbose]
```

Returns

List of dsp cell objects.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

XDC

Description

Returns a list of all DSP cell objects that have been declared in the current design.

The list of DSPs returned by `all_dsps` can also be limited or reduced by the `filter` command to filter according to properties assigned to the DSP objects. Properties of an object can be returned by the `list_property` or `report_property` commands.

The `all_dsps` command is scoped to return the objects hierarchically, from the top-level of the design or from the level of the current instance. By default the current instance is defined as the top level of the design, but can be changed by using the `current_instance` command.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Send Feedback

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example returns a list of all DSPs defined in the current design, and filters that list to return a single DSP assigned to the specified SITE:

```
filter [all_dsps] {SITE == DSP48_X1Y6}
```

The following example shows how the list returned can be passed to another command:

```
set_false_path -from [all_dsps] -to [all_registers]
```

See Also

- all_cpus
- all_hsios
- all_registers
- current_instance
- filter
- get_cells
- list_property
- report_property
- set_false_path

all_fanin

Get a list of pins or cells in fanin of specified sinks

Syntax

```
all_fanin [-startpoints_only] [-flat] [-only_cells] [-levels <arg>]
    [-pin_levels <arg>] [-trace_arcs <arg>] [-quiet] [-verbose] <to>
```

Returns

List of cell or pin objects.

Usage

Name	Description
[-startpoints_only]	Find only the timing startpoints
[-flat]	Hierarchy is ignored
[-only_cells]	Only cells
[-levels]	Maximum number of cell levels to traverse:Value >= 0 Default: 0
[-pin_levels]	Maximum number of pin levels to traverse:Value >= 0 Default: 0
[-trace_arcs]	Type of network arcs to trace: Values: timing, enabled, all
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<to>	List of sink pins, ports, or nets

Categories

[XDC](XDC)

Description

Returns a list of port, pin or cell objects in the fan-in of the specified sinks.

The `all_fanin` command is scoped to return objects from current level of the hierarchy of the design, either from the top-level or from the level of the current instance. By default the current instance is defined as the top level of the design, but can be changed by using the `current_instance` command. To return the fan-in across all levels of the hierarchy, use the `-flat` option.

Send Feedback

Arguments

`-startpoints_only` - (Optional) Find only the timing start points. When this option is used, none of the intermediate points in the fan-in network are returned. This option can be used to identify the primary driver(s) of the sinks.

`-flat` - (Optional) Ignore the hierarchy of the design. By default, only the objects at the same level of hierarchy as the sinks are returned. When using this option, all the objects in the fan-in network of the sinks are considered, regardless of hierarchy.

`-only_cells` - (Optional) Return only the cell objects which are in the fan-in path of the specified sinks. Do not return pins or ports.

`-levels <value>` - (Optional) Maximum number of cell levels to traverse. A value of 1 means to traverse the top-level of the current instance. The default value is 0 and indicates that the tool should traverse all levels in the hierarchy.

`-pin_levels <value>` - (Optional) Maximum number of pin levels to traverse. The default value is 0.

`-trace_arcs <value>` - (Optional) Type of network arcs to trace. Valid values are "timing", "enabled", and "all"

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<to> - (Required) The pins, ports, or nets from which you want the fan-in objects reported.

Examples

The following example lists the timing fan-in of the led_pins output port:

```
all_fanin [get_ports led_pins[*] ]
```

The following example traces back from the clock pin of the specified flip- flop to the clock source (an MMCM output pin in this example):

```
all_fanin -flat -startpoints_only [get_pins cmd_parse_i0/prescale_reg[7]/C]
```

Send Feedback

The following examples returns the ports connected to the input pins of IDELAYs, ignoring the hierarchy of the design:

```
all_fanin -flat -startpoints_only [get_pins IDELAY*/IDATAIN]
```

See Also

- all_fanout
- current_instance
- get_cells
- get_pins
- get_ports

Send Feedback

all_fanout

Get a list of pins or cells in fanout of specified sources

Syntax

```
all_fanout [-endpoints_only] [-flat] [-only_cells] [-levels <arg>]
    [-pin_levels <arg>] [-trace_arcs <arg>] [-quiet] [-verbose] <from>
```

Returns

List of cell or pin objects.

Usage

Name	Description
[-endpoints_only]	Find only the timing endpoints
[-flat]	Hierarchy is ignored
[-only_cells]	Only cells
[-levels]	Maximum number of cell levels to traverse:Value >= 0 Default: 0
[-pin_levels]	Maximum number of pin levels to traverse:Value >= 0 Default: 0
[-trace_arcs]	Type of network arcs to trace: Values: timing, enabled, all
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<from>	List of source pins, ports, or nets

Categories

[XDC](XDC)

Description

Returns a list of port, pin, or cell objects in the fanout of the specified sources.

The `all_fanout` command is scoped to return objects from current level of the hierarchy of the design, either from the top-level or from the level of the current instance. By default the current instance is defined as the top level of the design, but can be changed by using the `current_instance` command. To return the fanout across all levels of the hierarchy, use the `-flat` option.

Arguments

`-endpoints_only` - (Optional) Find only the timing endpoints. When this option is used, none of the intermediate points in the fan-out network are returned. This option can be used to identify the primary loads of the drivers.

`-flat` - (Optional) Ignore the hierarchy of the design. By default, only the objects at the same level of hierarchy as the sinks are returned. When using this option, all the objects in the fan-out network of the drivers are considered, regardless of hierarchy.

`-only_cells` - (Optional) Return only the cell objects in the fanout path of the specified sources.

`-levels <value>` - (Optional) Maximum number of cell levels to traverse. A value of 1 means to traverse the top-level of the current instance. The default value is 0 and indicates that the tool should traverse all levels in the hierarchy.

`-pin_levels <value>` - (Optional) Maximum number of pin levels to traverse. The default value is 0.

`-trace_arcs <value>` - (Optional) Type of network arcs to trace. Valid values are "timing", "enabled", and "all"

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<from>` - (Required) The source ports, pins, or nets from which to list the objects in the fanout path.

Examples

The following example gets the fanout for all input ports in the design:

```
all_fanout [all_inputs]
```

This example gets the fanout for all inputs assigned to I/O Bank 15 in the current design:

```
all_fanout [filter [all_inputs] {IOBANK == 15}]
```

See Also

- all_fanin
- current_instance
- filter
- get_cells
- get_pins
- get_ports

Send Feedback

all_ffs

Get a list of flip flop cells in the current design

Syntax

```
all_ffs [-quiet] [-verbose]
```

Returns

List of flip flop cell objects.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[XDC](XDC)

Description

Returns a list of all flip flop instances in the current design.

You can use the `get_cells` command, or use the `filter` command to limit the results from the `all_ffs` command to return a list of flip-flop cells matching the specified properties.

The `all_ffs` command is scoped to return the objects hierarchically, from the top-level of the design or from the level of the current instance. By default the current instance is defined as the top level of the design, but can be changed by using the `current_instance` command.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example returns the count of all flops in the design, then returns the count of all flops in the fftEngine module:

```
current_instance
INFO: [Vivado 12-618] Current instance is the top level of design
'netlist_1'.
top
llength [all_ffs]
15741
current_instance fftEngine
fftEngine
llength [all_ffs]
1519
```

This example filters the results of `all_ffs` to return only the FDRE flops:

```
filter [all_ffs] {REF_NAME == FDRE}
```

See Also

- all_latches
- all_registers
- current_instance
- filter
- get_cells

all_hsios

Get a list of hsio cells in the current design

Syntax

```
all_hsios [-quiet] [-verbose]
```

Returns

List of hsio cell objects.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[XDC](#)

Description

Returns a list of all High Speed IO (HSIO) cell objects that have been declared in the current design. These HSIO cell objects can be assigned to a variable or passed into another command.

The list of high-speed IOs returned by `all_hsios` can also be limited or reduced by the `filter` command to filter according to properties assigned to the HSIO objects. Properties of an object can be returned by the `list_property` or `report_property` commands.

The `all_hsios` command is scoped to return the objects hierarchically, from the top-level of the design or from the level of the current instance. By default the current instance is defined as the top level of the design, but can be changed by using the `current_instance` command.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Send Feedback

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example returns all HSIO objects in the current design:

```
all_hsios
```

The following example shows how the list returned can be directly passed to another command:

```
set_false_path -from [all_hsios] -to [all_registers]
```

See Also

- all_cpus
- all_dsps
- all_registers
- current_instance
- filter
- get_cells
- list_property
- report_property
- set_false_path

Send Feedback

all_inputs

Get a list of all input ports in the current design

Syntax

```
all_inputs [-quiet] [-verbose]
```

Returns

List of port objects.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

SDC, XDC

Description

Returns a list of all input port objects in the current design.

To get a list of specific inputs in the design, use the `get_ports` command, or use the `filter` command to filter the results returned by `all_inputs`.

The `all_inputs` command is scoped to return the objects hierarchically, from the top-level of the design or from the level of the current instance. By default the current instance is defined as the top level of the design, but can be changed by using the `current_instance` command.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example returns all input port objects in the current design:

```
all_inputs
```

This example gets all input port objects, filters out the GT ports, and sets the IOSTANDARD property for the non-GT ports:

```
set non_gt_ports [filter [all_inputs] {!is_gt_term}]
set_property IOSTANDARD LVCMOS18 $non_gt_ports
```

The following example shows how the list returned can be passed to another command:

```
set_input_delay 5 -clock REFCLK [all_inputs]
```

See Also

- all_clocks
- all_outputs
- current_instance
- filter
- get_clocks
- get_ports
- set_input_delay
- set_property

Send Feedback

all_latches

Get a list of all latch cells in the current design

Syntax

```
all_latches [-quiet] [-verbose]
```

Returns

List of latch cell objects.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[XDC](#)

Description

Returns a list of all latches that have been declared in the current design.

The list of latches returned by `all_latches` can also be limited or reduced by the `filter` command to filter according to properties assigned to the latches. Properties of an object can be returned by the `list_property` or `report_property` commands.

The `all_latches` command is scoped to return the objects hierarchically, from the top-level of the design or from the level of the current instance. By default the current instance is defined as the top level of the design, but can be changed by using the `current_instance` command.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Send Feedback

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example returns a list of all latches in the current design:

```
all_latches
```

The following example shows how the list returned can be passed to another command:

```
set_false_path -from [all_mults] -to [all_latches]
```

See Also

- all_ffs
- all_registers
- current_instance
- filter
- get_cells
- list_property
- report_property
- set_false_path

all_outputs

Get a list of all output ports in the current design

Syntax

```
all_outputs [-quiet] [-verbose]
```

Returns

List of port objects.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[SDC](), [XDC]()

Description

Returns a list of all output port objects that have been declared in the current design.

To get a list of specific outputs in the design, use the `get_ports` command, or use the `filter` command to filter the results returned by `all_outputs`.

The `all_outputs` command is scoped to return the objects hierarchically, from the top-level of the design or from the level of the current instance. By default the current instance is defined as the top level of the design, but can be changed by using the `current_instance` command.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Send Feedback

Examples

The following example returns all the output ports in the current design:

```
all_outputs
```

The following example sets the output delay for all outputs in the design:

```
set_output_delay 5 -clock REFCLK [all_outputs]
```

See Also

- all_inputs
- current_instance
- filter
- get_ports
- set_output_delay

all_rams

Get a list of ram cells in the current design

Syntax

```
all_rams [-quiet] [-verbose]
```

Returns

List of ram cell objects.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[XDC]

Description

Returns a list of all the RAM cell objects present in the current instance, including Block RAMS, Block RAM FIFOs, and Distributed RAMS. These RAM cell objects can be assigned to a variable or passed into another command.

To get a list of specific RAM cells in the design, use the `filter` command to filter the results returned by `all_rams` based on properties assigned to the RAM cells. Properties of an object can be returned by the `list_property` or `report_property` commands.

The `all_rams` command is scoped to return the objects hierarchically, from the top-level of the design or from the level of the current instance. By default the current instance is defined as the top level of the design, but can be changed by using the `current_instance` command.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example returns all RAM cells in the design:

```
all_rams
```

This example returns all RAM cells in the design, and filters the results to return only the FIFO block memories:

```
filter [all_rams] {PRIMITIVE_SUBGROUP == fifo}
```

The following example sets the current instance, and returns all RAM objects hierarchically from the level of the current instance:

```
current_instance usbEngine0
all_rams
```

See Also

- all_cpus
- current_instance
- filter
- get_cells
- list_property
- report_property

all_registers

Get a list of register cells or pins in the current design

Syntax

```
all_registers [-clock <args>] [-rise_clock <args>] [-fall_clock <args>]
    [-cells] [-data_pins] [-clock_pins] [-async_pins] [-output_pins]
    [-level_sensitive] [-edge_triggered] [-no_hierarchy] [-quiet]
    [-verbose]
```

Returns

List of cell or pin objects.

Usage

Name	Description
[-clock]	Consider registers of this clock
[-rise_clock]	Consider registers triggered by clock rising edge
[-fall_clock]	Consider registers triggered by clock falling edge
[-cells]	Return list of cells (default)
[-data_pins]	Return list of register data pins
[-clock_pins]	Return list of register clock pins
[-async_pins]	Return list of async preset/clear pins
[-output_pins]	Return list of register output pins
[-level_sensitive]	Only consider level-sensitive latches
[-edge_triggered]	Only consider edge-triggered flip-flops
[-no_hierarchy]	Only search the current instance
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

SDC, XDC

Description

Returns a list of sequential register cells or register pins in the current design.

> 💡 **TIP:** *Returned objects includes DSPs and BRAMs as they contain internal registers.*

The list of returned objects can be limited by the use of the arguments described below. You can limit the list of registers returned to a specific clock or clocks, or to registers triggered by the rising or falling edge of a specified clock.

The list of registers returned by `all_registers` can also be limited or reduced by the `filter` command to filter according to properties assigned to the registers. Properties of an object can be returned by the `list_property` or `report_property` commands.

You can also get a list of the pins of collected registers instead of the register objects by specifying one or more of the pin arguments.

Arguments

`-clock` <args> - (Optional) Return a list of all registers whose clock pins are in the fanout of the specified clock.

`-rise_clock` <args> - (Optional) Return a list of registers triggered by the rising edge of the specified clocks.

`-fall_clock` <args> - (Optional) Return a list of registers triggered by the falling edge of the specified clocks.

Note: Do not combine `-clock`, `-rise_clock`, and `-fall_clock` in the same command.

`-cells` - (Optional) Return a list of register cell objects as opposed to a list of pin objects. This is the default behavior of the command.

`-data_pins` - (Optional) Return a list of data pins of all registers in the design, or of the registers that meet the search requirement.

`-clock_pins` - (Optional) Return a list of clock pins of the registers that meet the search requirement.

`-async_pins` - (Optional) Limit the search to asynchronous pins of the registers that meet the search requirement.

`-output_pins` - (Optional) Return a list of output pins of the registers that meet the search requirement.

Note: Use the `-*_pins` arguments separately. If you specify multiple arguments, only one argument is applied in the following order of precedence: `-data_pins`, `-clock_pins`, `-async_pins`, `-output_pins`.

`-level_sensitive` - (Optional) Return a list of the level-sensitive registers or latches.

`-edge_triggered` - (Optional) Return a list of the edge-triggered registers or flip-flops.

`-no_hierarchy` - (Optional) Do not search the hierarchy of the design. Only search in the level of the `current_instance`.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example returns a list of registers that are triggered by the falling edge of any clock in the design:

```
all_registers -fall_clock [all_clocks]
```

The following example sets the minimum delay:

```
set_min_delay 2.0 -to [all_registers -clock CCLK -data_pins]
```

The following example extracts all registers on clk_A with *meta* in the name:

```
filter [all_registers -clock clk_A] {name =~ *meta*}
```

See Also

- all_clocks
- current_instance
- filter
- list_property
- report_property
- set_msg_config
- set_min_delay

Send Feedback

apply_bd_automation

Runs an automation rule on an IPI object.

Syntax

```
apply_bd_automation -rule <arg> [-config <args>] -dict <arg> -opts <arg>
    [-quiet] [-verbose] <objects>...
```

Returns

Returns success or failure.

Usage

Name	Description
-rule	Rule ID string
[-config]	List of parameter value pairs
-dict	List of objects and corresponding parameter name-value pairs.
-opts	List of settings that apply to all objects in this rule.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<objects>	The objects to run the automation rule on

Categories

IPIntegrator

Description

IP integrator provides the Designer Assistance feature, using the `apply_bd_automation` command, to automatically configure and/or add other relevant IP integrator cells around a selected IP integrator object. For more information on the Designer Assistance features refer to the *Vivado Design Suite User Guide: Designing with IP* (UG896) or the *Vivado Design Suite User Guide: Designing IP Subsystems Using IP Integrator* (UG994).

Currently block and connection automation exists for cells, interfaces, pins and ports. The Block Automation feature is provided for certain complex blocks such as the Zynq device, MicroBlaze processor, AXI Ethernet and memory IP.

The Connection Automation feature helps automate different types of connections. For instance when connecting Slave AXI-MM interfaces, the automation will also connect up the relevant clock and reset pins and also create an interconnect if one is required. Connection Automation may also help with board-level connections; connecting pins and interfaces from relevant cells, to external ports and interfaces, and applying appropriate board constraints on these external I/Os.

Note: This IP integrator command is issued from within the Vivado IDE via the Designer Assistance GUI feature. It is recommended that you make use of this command in IP integrator through the Vivado IDE, rather than directly from Tcl scripts. Use the `write_bd_tcl` command to output Tcl for use within a user script.

Arguments

`-rule <arg>` - (Required) Specify the defined automation rules to use for the selected object.

`-config <args>` - (Optional) Specify a list of configuration parameters for the IP integrator object, and the value to assign to the parameter. The parameter name (`param`) is provided without quotes, while the value is quoted to distinguish it from the param. The parameter and value are assigned as a pair (`param "value"`). Multiple `param "value"` pairs can be defined for the specified `<objects>`, and should be enclosed in braces, {}.

```
-config {local_mem "16KB" ecc "Basic" debug_module "Debug Only"}
```

`-dict` - (Optional) Use this option to specify a dictionary of block design objects with configuration parameters for each object, specified as `<object> <params>` pairs.

```
apply_bd_automation -rule <ruleID> \
    -dict "[get_bd_intf_net /intf_net0] { DATA_SEL "1" TRIG_SEL "2" } \
    [get_bd_intf_net /intf_net1] { DATA_SEL "2" } \
    [get_bd_net /net1]  {WIDTH "32" TYPE "1" }"
```

> **TIP:** *Use the `-dict` option in place of the `<objects>` and `-config` arguments when specifying multiple values.*

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<objects>` - (Required) The IP integrator object to apply automation to. The object must be returned by commands like `get_bd_cells`, `get_bd_pins`, and `get_bd_interface`, and cannot simply be referenced by name. Only a single IP integrator object should be specified, and it must be compatible with the specified `-rule` and `-config` options.

Example

The following example applies Block Automation to the MicroBlaze cell according to the specified rules, configuring the specified parameters with the provided values:

```
apply_bd_automation -rule xilinx.com:bd_rule:microblaze \
    -config {local_mem "16KB" ecc "Basic" debug_module "Debug Only" \
    axi_periph "1" axi_intc "1" clk "New Clocking Wizard (100 MHz)" }  \
    [get_bd_cells /microblaze_1]
```

In this example, the Connection Automation feature applies the board rules to an IP subsystem pin or interface, in this case the clock interface object, when a known compatible interface is available on the board. The first command, `get_board_interfaces`, returns the interfaces on the target board that are compatible with the IP object. The second command, `apply_bd_automation`, connects the clock interface to the selected board interface:

```
get_board_interfaces -filter "VLNV==[get_property VLNV \
    [get_bd_intf_pins clk_wiz_1/CLK_IN1_D]]"
sys_diff_clock
apply_bd_automation -rule xilinx.com:bd_rule:board \
    -config {Board_Interface "sys_diff_clock" } \
    [get_bd_intf_pins /clk_wiz_1/CLK_IN1_D]
```

This example applies custom board rules to the IP subsystem clock interface object, CLK_IN1_D, when a clock interface is not available on the target board:

```
apply_bd_automation -rule xilinx.com:bd_rule:board \
    [get_bd_pins /clk_wiz_2/CLK_IN1_D]
```

This example applies custom board rules to an IP subsystem reset_pin, ext_reset_in, when the reset interface on the board is not available:

```
apply_bd_automation -rule xilinx.com:bd_rule:board \
    -config {rst_polarity "ACTIVE_HIGH" } \
    [get_bd_pins /proc_sys_reset_1/ext_reset_in]
```

See Also

- create_bd_cell
- create_bd_design
- get_bd_cells
- get_bd_intf_pins
- get_bd_pins
- get_board_interface_ports

- get_board_part_interfaces
- write_bd_tcl

apply_board_connection

Applies board connections to given designs

Syntax

```
apply_board_connection [-board_interface <arg>] -ip_intf <arg>
    -diagram <arg> [-quiet] [-verbose]
```

Returns

Sucess/failure status of applied action.

Usage

Name	Description
[-board_interface]	Board interface name to which connection need to be applied.
-ip_intf	Full path of IP interface name to which board automation need to be applied.
-diagram	The IP Integrator design name.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Object, Project, Board, IPIntegrator

Description

Connects the interface pin of an IP core in the specified block design to the interface of the current board part in the current project or design.

The board part provides a representation of the Xilinx device in the context of the board-level system, and can help define key aspects of the design, such as clock constraints, I/O port assignments, and supported interfaces. The board part file stores information regarding board attributes. The file, called `board_part.xml`, is located in the `data/boards/board_parts` folder of the Vivado Design Suite installation area.

The command lets you quickly connect compatible interface pins of an IP integrator block design to the appropriate interface definition on the current board part. To make the connection between the IP core to the board part, the IP integrator of the Vivado Design Suite adds an external interface port and interface connection to the block design. The added external interface port is named for the specified board part interface.

Send Feedback

The `apply_board_connection` commands uses the available interfaces of the current board part defined in the project. An error is returned if the project uses a target part rather than a target board. You can use the `current_board_part` command to identify the target board used by the project, or `get_board_parts` to list the boards available for use by the project. Use the `get_board_part_interfaces` command to determine the list of available interfaces on the current board.

To remove an existing IP interface connection, specify the `-ip_intf` option, but do not specify the `-board_interface`. If no board part interface is specified, the IP interface pin is disconnected.

This command returns a transcript of it actions, or returns an error if it fails.

Arguments

`-board_interface <arg>` - (Optional) Specifies the interface definition to connect to the specified IP core interface. If the `-board_interface` option is not specified, or is specified with an empty string value, "", an existing connection on the IP interface pin will be removed.

`-ip_intf <arg>` - (Required) Specifies the interface pin of an IP core in the IP integrator Block Design defined by the `-diagram` option. The IP Interface pin is specified in the form `<IP_core_name>/<interface_pin_name>`. In the first example below, the IP core instance name is `"/proc_sys_reset_0"` and the Interface pin is `"/ext_reset"`, combined as seen in the example.

`-diagram <arg>` - (Required) The name of the IP integrator block design where the IP core instance is found.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Example

The following example connects the SGMII interface, defined in the current board part, to the processor reset IP core:

```
apply_board_connection -board_interface "reset" \
    -ip_intf "/proc_sys_reset_0/ext_reset" -diagram "design_2"
```

This example removes the connection on the specified interface pin, because the `-board_interface` option is an empty string:

```
apply_board_connection -board_interface "" \
    -ip_intf "/proc_sys_reset_0/ext_reset" -diagram "design_2"
```

See Also

- create_port
- current_board_part
- get_board_part_interfaces
- get_board_parts
- write_checkpoint
- write_edif

apply_hw_ila_trigger

Apply trigger at startup init values to an ILA core in the design

Syntax

```
apply_hw_ila_trigger [-ila_cell <arg>] [-quiet] [-verbose] [<file>]
```

Usage

Name	Description
[-ila_cell]	Apply trigger settings to this ila cell
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<file>]	ILA startup trigger settings file

Categories

[Debug](#)

Description

Apply a trigger configuration file to the bitstream of a design, to support ILA trigger at startup.

This command is used to configure the trigger settings of an ILA core in a design bitstream (.bit) file, so that the ILA debug core is armed to trigger on events immediately after device configuration and startup. This allows data to be captured from the earliest stages of device activity, which would not be possible through the use of the Hardware Manager feature of the Vivado Design Suite, and the `run_hw_ila` command.

The `apply_hw_ila_trigger` command reads a trigger configuration file written by `run_hw_ila -file` and applies the various trigger settings to the ILA core in the implemented design. The trigger configuration for the ILA core then become part of the bitstream written by `write_bitstream`, that is used to program the Xilinx FPGA device.

The process for using the trigger at startup feature includes the following steps:

1. From the Hardware Manager, use `run_hw_ila -file` to export the trigger register map file for the ILA core.

2. Open the implemented design, or the implemented design checkpoint.

3. Use the `apply_hw_ila_trigger` command to apply the trigger settings to the in-memory design.

4. Use the `write_bitstream` command to write the bitstream with the applied trigger configuration file.

   Note: Be sure to use the `write_bitstream` command, and not the Flow Navigator commands in the Vivado IDE.

5. Return to the Hardware Manager, and use `program_hw_device` to program the hw_device using the new bitstream file.

Once programmed, the new ILA core should immediately arm at startup. In the Vivado logic analyzer feature, you should see the "Trigger Capture Status" for the ILA core is now populated with captured data samples if trigger events or capture conditions have occurred. Refer to the *Vivado Design Suite User Guide: Vivado Programming and Debugging (UG908)* for more information.

Arguments

`-ila_cell <arg>` - (Optional) Apply trigger settings to the specified ILA cell. The cell must be specified as an object using the `get_cells` command.

Note: The trigger configuration file includes the instance information for the ILA cell it applies to. The `-ila_cell` option is not required, and should only be used to apply the trigger file to a different ILA cell in the design.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file>` - (Optional) Specify a trigger configuration file that was created by the `run_hw_ila -file` command.

Example

The following example applies the trigger configurations to the ILA cell instance that is defined in the specified file:

```
apply_hw_ila_trigger C:/Data/ila1_triggers.tas
```

See Also

- current_hw_device
- current_hw_ila

Send Feedback

- get_hw_ilas
- program_hw_devices
- run_hw_ila
- write_bitstream

Send Feedback

Chapter 3: Tcl Commands Listed Alphabetically

archive_project

Archive the current project

Syntax

```
archive_project [-temp_dir <arg>] [-force] [-exclude_run_results]
    [-include_config_settings] [-include_runs_in_progress]
    [-include_local_ip_cache] [-quiet] [-verbose] [<file>]
```

Returns

True

Usage

Name	Description
[-temp_dir]	specify temporary location to save project copy to archive Default: .
[-force]	Overwrite existing archived file
[-exclude_run_results]	Exclude run results from the archive
[-include_config_settings]	Include current project environment configuration settings/files in archive
[-include_runs_in_progress]	Include run result even if the run is in progress, this switch will be ignored if -exclude_run_results is specified
[-include_local_ip_cache]	Include IP cache results in the archive
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<file>]	Name of the archive file

Categories

Project

Description

Archives a project to store as backup, or to encapsulate the design and send it to a remote site. The tool parses the hierarchy of the design, copies the required source files, include files, and remote files from the library directories, copies the constraint files, copies the results of the various synthesis, simulation, and implementation runs, and then creates a ZIP file of the project.

> 💡 **TIP:** *In order to archive the tcl.pre and tcl.post scripts, associated with the synthesis and implementation steps in the Design Run Settings dialog box, you must add these script files to the project as design sources.*

An alternative method of archiving the project is using `write_project_tcl` to create a Tcl script that will recreate the project in its current form.

UG835 (v2021.1) June 16, 2021
Tcl Command Reference Guide

Send Feedback

www.xilinx.com
110

Arguments

`-temp_dir <arg>` - (Optional) Specify a temporary directory to copy files to when creating the project archive. The temporary directory will be created if it does not exist, and will be emptied when the archive process is complete. By default, the Vivado tool will create a temporary directory inside of the current working directory.

`-force` - (Optional) Overwrite an existing ZIP file of the same name. If the ZIP file exists, the tool returns an error unless the `-force` argument is specified.

`-exclude_run_results` - (Optional) Exclude the results of any synthesis or implementation runs. This command can greatly reduce the size of a project archive.

`-include_config_settings` - (Optional) Add any initialization Tcl commands (`init.tcl`) to the archive file under the `<project_name>/config_settings` folder.

`-include_runs_in_progress` - (Optional) Include data in the archive from the directories of incomplete runs. This switch will be ignored if `-exclude_run_results` is specified.

`-include_local_ip_cache` - (Optional) Include the cached IP synthesis results in the project archive. This command only includes cached synthesis results if the IP cache is local to the project rather than remote. Refer to the `config_ip_cache` command for more information.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file>` - (Optional) The name of the ZIP file to be created by the `archive_project` command. If `<file>` is not specified, a ZIP file with the same name as the project is created.

Examples

The following command archives the current project:

```
archive_project
```

Note: The project archive is named `<project_name>.zip` because no file name is specified.

The following example specifies `project_3` as the current project, and then archives that project into a file called `proj3.zip`:

```
current_project project_3
archive_project -force -exclude_run_results proj3.zip
```

Note: The use of the `-force` argument causes the tool to overwrite the `proj3.zip` file if one exists. The use of the `-exclude_run_results` argument causes the tool to leave any results from synthesis or implementation runs out of the archive. The various runs defined in the project are included in the archive, but not any of the results.

The following command archives the current project in the specified file, overwrites an existing file if needed, excludes the run results, and includes any configuration settings used when launching the Vivado tool:

```
archive_project -force mb1_archive.zip -temp_dir C:/Data/Temp \
    -exclude_run_results -include_config_settings
```

See Also

- current_project
- write_project_tcl

Send Feedback

assign_bd_address

Automatically assign addresses to unmapped IP

Syntax

```
assign_bd_address [-target_address_space <arg>] [-boundary]
    [-master_boundary] [-external] -dict <arg> [-import_from_file <arg>]
    [-export_to_file <arg>] [-export_gui_to_file <arg>] [-offset <arg>]
    [-range <arg>] [-base_high <arg>] [-force] [-quiet] [-verbose]
    [<objects>...]
```

Returns

The newly mapped segments, "" if failed.

Usage

Name	Description
[-target_address_space]	Target address space to place segment into
[-boundary]	assign peripherals to the exported slave hierarchical boundary of the design
[-master_boundary]	set hierarchical master boundary by assigning exported master interface segments to internal masters
[-external]	allow an external master interface to be mapped to more than one address
-dict	dictionary of offset range address pairs, e.g. {offset 0x00000000 range 32K offset 0x20000000 range 32K} used to map an external interface to more than one address
[-import_from_file]	import the addressing from file. When extension is csv, format = <address space name>,<slave segment>,<offset>,<range>
[-export_to_file]	export the address map to a file in csv format. format = <address space name>,<slave segment>,<offset>,<range>
[-export_gui_to_file]	export the address map to a file in csv format. Structure is same as Address Editor GUI
[-offset]	Offset of assignment. e.g. 0x00000000
[-range]	Range of assignment. e.g. 4096, 4K, 16M, 1G
[-base_high]	colon separated base:high offsets of a range assignment. e.g. 0x0000:0xFFFF
[-force]	force the assignment, do not run any validity checks
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<objects>]	The objects to assign

Categories

IPIntegrator

Description

Assign unmapped IP address segments to address spaces in the IP integrator subsystem design.

If the target address space is not specified, the IP integrator will automatically assign the address segment to an available address space on a connected AXI master.

If no bd_addr_seg objects are specified the `assign_bd_address` command will assign all unmapped address segments to any connected AXI master address spaces.

This command returns the newly mapped address segments, or returns an error if it failed.

Arguments

`-target_address_space <arg>` - (Optional) The target address space to place the segment into.

> **TIP:** *If the target address space is not specified, the Vivado IP integrator will automatically assign the address segment to an available address space.*

`-boundary` - (Optional) Assign address spaces to the hierarchical boundary of the design. This lets you allocate address space for memory that exists outside of the block design. Map scoped external slave interfaces into the address space at contiguous offsets beginning at 0x000_0000.

`-master_boundary` - (Optional) For the scoped external master interfaces, a slave segment is created is mapped into internal masters on the scoped diagram, at the specified `-offset` and `-range`.

`-external` - (Optional) Define memory segment of a previously defined external address space. Multiple memory segments can be assigned to an external AXI port that connects to a Slave IP outside of the BD.

`-import_from_file <arg>` - (Optional) Read address space assignments from a specified CSV format file containing a memory map. The CSV file must be in the form `<address space name>,<slave segment>,<offset>,<range>`. This command will overwrite any existing address space definition if the CSV file contains a valid address map. Otherwise the command will return an error.

`-export_to_file <arg>` - (Optional) Write address space assignments to a specified CSV format file. The CSV file will be in the form `<address space name>,<slave segment>,<offset>,<range>`. This command will not overwrite an existing file of the same name, unless `-force` is also specified.

`-force` - (Optional) Overwrite an existing CSV file when used with the `-export_to_file` option.

`-base_high <arg>` - (Optional) Assign the address space by specifying the <base address>:<high address>, in the form `-base_high 0x4000_0000:0x4000_FFFF` for instance.

`-offset <arg>` - (Optional) Specify the memory offset of address space to assign the address segment to. For example, 0x00000000.

`-range <arg>` - (Optional) Range, or size, of address space to assign the address segment into. For example 4096, 4K, 16M, 1G.

`-dict <arg>` - (Optional) A dictionary of Offset/Range value pairs to be defined at one time. The format of the offset/range value pair is as follows:

```
-dict {offset 0x00000000 range 64M offset 0x20000000 range 64M}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<objects>` - The block design address segment objects to assign to the target address space.

Example

The following example automatically assigns the specified address segment object to an address space. In this example, the target space is not provided, so the IP integrator automatically assigns the address segment to an available address space:

```
assign_bd_address [get_bd_addr_segs \
{/microblaze_1_local_memory/ilmb_bram_if_cntlr/SLMB/Mem }]
```

The following example creates an memory space on an external AXI port, and then assigns specified block design address segments to two separate address ranges, with specified offsets, within that external address space:

```
assign_bd_address -offset 0xA0000000 -range 64K [get_bd_addr_segs {/
M02_AXI_0/Reg}]
assign_bd_address -external -offset 0xA0000000 -range 4K \
[get_bd_addr_segs /M02_AXI_0/Seg] -target_address_space [get_bd_addr_space /
M02_AXI_0/Reg]
assign_bd_address -external -offset 0xA0001000 -range 4K \
[get_bd_addr_segs /M02_AXI_0/Seg] -target_address_space [get_bd_addr_space /
M02_AXI_0/Reg]
```

This example is the same as the prior example, except that it uses the `-dict` argument to specify multiple offset/range value pairs in a single `assign_bd_address` command:

```
assign_bd_address -offset 0xA0000000 -range 64K [get_bd_addr_segs {/
M02_AXI_0/Reg}]
assign_bd_address -external -dict {offset 0xA0000000 range 4K offset
0xA0001000 range 4K} \
[get_bd_addr_segs /M02_AXI_0/Seg] -target_address_space [get_bd_addr_space /
M02_AXI_0/Reg]
```

See Also

- create_bd_addr_seg
- exclude_bd_addr_seg
- get_bd_addr_segs
- get_bd_addr_spaces
- include_bd_addr_seg

Send Feedback

auto_detect_xpm

Auto detect the XPM Libraries that are used in the design and set the XPM_LIBRARIES project property.

Syntax

```
auto_detect_xpm [-quiet] [-verbose]
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

FileIO, Project

Description

This command collects the list of Xilinx Parameterized Macros (XPM) used in RTL design files and sets the XPM_LIBRARIES property. Because it identifies XPM used in the RTL source files, it must be used after the RTL files are read or added to the design as shown in the example.

> **TIP:** *This command is intended for use in non-project mode, and is not needed for project-based flows.*

This command will find all the XPMs used in the user RTL and update the XPM_LIBRARIES property on the current project. Synthesis and simulation will use the XPM_LIBRARIES property to properly support these macros.

This command returns nothing, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example reads the specified RTL file, and then uses `auto_detect_xpm` to define the XPM_LIBRARIES used in the files:

```
read_verilog ../top_bgRAM_173_0.v
read_xdc ../top_bgRAM_173_0.xdc
auto_detect_xpm
```

See Also

- read_verilog
- read_vhdl

boot_hw_device

Issue JTAG Program command to hw_device

Syntax

```
boot_hw_device [-disable_done_check] [-timeout <arg>] [-quiet] [-verbose]
    <hw_device>
```

Usage

Name	Description
`[-disable_done_check]`	Disable done check for boot device
`[-timeout]`	Time out for boot (seconds) Default: default
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<hw_device>`	Target hw_device connection

Categories

Hardware

Description

Issue JTAG PROGRAM command to the hw_device (FPGA).

The boot_hw_device command triggers the FPGA boot and board startup sequence. The boot sequence starts the FPGA configuration process to clear the device of any prior programming, and then to load a new program, depending on the mode pin settings.

The hw_device will boot based on its mode pin settings. If the FPGA's mode pins on the device are set to JTAG mode, or the interface is not active (e.g. the PROM is not configured) the net effect of the `boot_hw_device` command is to clear the prior programming.

This command returns a 1 if it detects that the DONE pin has gone HIGH, or when the device has been cleared, otherwise it returns 0.

Arguments

`-disable_done_check` - (Optional) Disable done check for boot device. Use this option when you intend to clear a programmed device without reprogramming it, in JTAG mode for instance.

`-timeout <arg>` - (Optional) Specify the timeout value in seconds for when the `boot_hw_device` command will stop trying to boot the device.

Send Feedback

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_device> - (Optional) Specify the hw_device object to send the JTAG boot command to. The hw_device must be specified as an object as returned by the `get_hw_devices` or `current_hw_device` commands.

Example

The following example creates a memory device and associates it with the `current_hw_device`, assigning various properties of the hw_cfgmem object. Then it uses the `write_cfgmem` command to format the design bitstream, and associate the cfgmem (`.mcs`) file with the PROGRAM.FILE property of the hw_cfgmem object. Then it programs the hw_cfgmem object with the data from the MCS file starting at the specified offset address. After programming, the `boot_hw_device` command is issued for the `current_hw_device` and the DONE pin is checked, and an appropriate message is returned:

```
set memObj [create_hw_cfgmem -hw_device [current_hw_device] \
    [lindex [get_cfgmem_parts {28f00ap30t-bpi-x16}] 0]]
set_property PROGRAM.BLANK_CHECK  1 $memObj
set_property PROGRAM.ERASE  1 $memObj
set_property PROGRAM.CFG_PROGRAM  1 $memObj
set_property PROGRAM.VERIFY  1 $memObj
write_cfgmem -force -format MCS -size 64 -interface BPIx16 \
    -loadbit "up 0x0 ./project_netlist.runs/impl_1/sinegen_demo.bit" \
    ./config_28f00ap30t
set_property PROGRAM.FILE ./config_28f00ap30t.mcs $memObj
program_hw_cfgmem -hw_cfgmem $memObj
if {[boot_hw_device [current_hw_device]] == 1} {
    puts stderr "DONE signal is HIGH"
} else {
    puts stderr "DONE signal is LOW"
}
```

See Also

- connect_hw_server
- create_hw_cfgmem
- current_hw_device
- current_hw_target
- get_hw_devices
- get_hw_targets

Send Feedback

- program_hw_cfgmem

calc_config_time

Calculate device configuration time (ms)

Syntax

```
calc_config_time [-verbose] [-max] [-min] [-typical] [-por_used]
    [-por_ramp <arg>] [-clk_freq <arg>] [-bitstream_size <arg>] [-quiet]
```

Returns

Report

Usage

Name	Description
[-verbose]	Print out calculation parameters
[-max]	Calculate Maximum Configuration Time
[-min]	Calculate Minimum Configuration Time
[-typical]	Calculate Typical Configuration Time
[-por_used]	(Deprecated) Specify if Power On Reset (POR) is used by using a non-zero por_ramp
[-por_ramp]	Specify a Power On Reset (POR) ramp rate as 1 ms to 50 ms Default: 0 ms
[-clk_freq]	Specify a clock frequency for Slave mode, or for Master mode if using external master clock (MHz) Default: 0 MHz
[-bitstream_size]	Specify a bitstream size to override the default Default: 0
[-quiet]	Ignore command errors

Categories

Report

Description

Estimates the time in milliseconds (ms) to configure a Xilinx device for the current design.

💡 **TIP:** *The Device Configuration Mode must be defined for this command to work.*

Send Feedback

Some applications require that the Xilinx device be configured and operational within a short time. This command lets you estimate the configuration time for the device and design in question. The configuration time includes the device initialization time plus the configuration time. Configuration time depends on the size of the device and speed of the configuration logic. For more information on the configuration time refer to *UltraFast Design Methodology Guide for the Vivado Design Suite* (UG949), the *UltraScale Architecture Configuration User Guide* (UG570), or the *7 Series FPGAs Configuration User Guide* (UG470).

Some of the settings needed to calculate the configuration time are stored as properties on the current design, such as the BITSTREAM.CONFIG.CONFIGRATE or BITSTREAM.CONFIG.EXTMASTERCCLK_EN properties. In some master modes, the FPGA provides the configuration clock to control configuration, with the nominal configuration clock frequency specified by BITSTREAM.CONFIG.CONFIGRATE. The property can be defined in the **Edit Device Properties** dialog box of the Vivado Design Suite IDE, or by using `set_property` to directly set the value of the specified property.

For a slave configuration mode, or for configuration modes using an external master clock, the needed clock frequency is specified by the `-clk_freq` option.

This command returns a value in milliseconds if successful, or returns an error if it fails.

Arguments

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`-max` - (Optional) Reports the maximum configuration time as estimated by the command.

`-min` - (Optional) Reports the minimum configuration time as estimated by the command.

`-typical` - (Optional, default) Reports the typical configuration time as estimated by the command.

`-por_ramp <arg>` - (Optional) Specify the power-on reset (POR) ramp rate as a value from 1 millisecond (ms) to 50 ms. The default is 0. You can reduce the power on reset time (Tpor) by controlling the ramp rate on the system. For specifications on the Tpor ramp options, see the FPGA configuration switching characteristics in the data sheet of the specific device in use.

`-clk_freq <arg>` - (Optional) Specify a clock frequency in MHz for Slave mode, or when using an external master clock. The default is 0.

Note: When run in a master configuration mode, the clock frequency is specified through the BITSTREAM.CONFIG.CONFIGRATE property.

`-por_used <arg>` - (Deprecated) Specify if a Power On Reset (POR) is used. Replaced by using a non-zero `por_ramp` command line option.

Send Feedback

`-bitstream_size <arg>` - (Optional) Specify a bitstream size in bits. Default is the design bitstream size.

> **TIP:** *The bitstream size is derived automatically from the bitstream associated with the current design. Specifying this option will override the automatically derived value.*

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

Example

The following example returns the maximum calculated configuration time for the current target part based on the specified external clock frequency:

```
calc_config_time -max -clk_freq 50
```

See Also

- program_hw_cfgmem
- set_property
- write_bitstream

Send Feedback

can_resolve_reference

Check if a module can be referenced

Syntax

```
can_resolve_reference [-quiet] [-verbose] <module>...
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<module>	module name

Categories

Object, Project

Description

This command is used to validate reference to modules prior to trying to import them into the design. Its primary use is in scripts like the script produced by the `write_bd_tcl` command, though you can use it in your own scripts as well.

This command returns 0 if the reference cannot be resolved, or returns 1 if it can be resolved.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<module> - (Required) Specifies the name of the module to validate. The module name is referenced from a loaded RTL design source file.

Examples

This example determines if the reference to the specified module can be resolved:

```
can_resolve_reference clk_div
```

See Also

- [write_bd_tcl](#)
- [write_project_tcl](#)

check_syntax

Check HDL syntax in the supplied fileset or active fileset.

Syntax

```
check_syntax [-fileset <arg>] [-return_string] [-quiet] [-verbose]
```

Usage

Name	Description
[-fileset]	Fileset to check for syntax
[-return_string]	Return the syntax check messages as a string
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Project

Description

Parses Verilog, SystemVerilog, and VHDL source files and generates syntax warnings and error messages for the design.

> **TIP:** *The syntax is also checked automatically as the file is edited in the Vivado text editor, or when the file is saved.*

This command returns warnings or errors related to the files it examines, or returns nothing if no problems are found.

Arguments

`-fileset <arg>` - (Optional) Check the syntax of files in the specified fileset.

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example checks the syntax of files in the simulation fileset:

```
check_syntax -fileset sim_1
```

See Also

• get_filesets

check_timing

Check the design for possible timing problems

Syntax

```
check_timing [-file <arg>] [-no_header] [-loop_limit <arg>] [-append]
    [-name <arg>] [-override_defaults <args>] [-include <args>]
    [-exclude <args>] [-return_string] [-rpx <arg>] [-cells <args>]
    [-verbose] [-quiet]
```

Usage

Name	Description
[-file]	Filename to output results to. (send output to console if -file is not used)
[-no_header]	do not generate a report header
[-loop_limit]	Limit the number of loops reported for loops check Default: 100
[-append]	Append the results to file, don't overwrite the results file
[-name]	Output the results to GUI panel with this name
[-override_defaults]	Overrides the checks in the default timing checks listed below
[-include]	Add this list of checks to be performed along with default timing checks listed below
[-exclude]	Exclude this list of checks to be performed from the default timing checks listed below
[-return_string]	return report as string
[-rpx]	Filename to output interactive results to.
[-cells]	run check_timing on the specified cell(s)
[-verbose]	Return a detailed list of all timing problems found
[-quiet]	Ignore command errors

Categories

[Report](), [Timing]()

Description

Checks the design elements of ports, pins, and paths, against the current timing constraints. Use this command to identify possible problems with design data and timing constraints before running the `report_timing` command. The `check_timing` command runs a series of default timing checks, and reports a summary of any violations found. To get detailed information about violations, use the `-verbose` option.

Note: By default the report is written to the Tcl console or STD output. However, the results can also be written to the GUI with the `-name` option, or to a file with `-file`.

Default Timing Checks:

- `constant_clock` - Checks for clock signals connected to a constant signal (gnd/vss/data).

- `generated_clocks` - Checks for loops, or circular definitions within the generated clock network. This check will return an error if a generated clock uses a second generated clock as its source, when the second generated clock uses the first clock as its source.

- `latch_loops` - Checks for and warns of combinational latch loops in the design.

- `loops` - Checks for and warns of combinational feedback loops in the design.

- `multiple_clock` - Warns if multiple clocks reach a register clock pin. If more than one clock signal reaches a register clock pin it is unclear which clock will be used for analysis. In this case, use the `set_case_analysis` command so that only one clock will propagate to the register clock pin.

- `no_clock` - Reports unclocked registers. In this case, no setup or hold checks are performed on data pins related to the register clock pin.

- `no_input_delay` - Reports the input ports without an input delay constraint. Input delays can be assigned using the `set_input_delay` command. Input ports that are unclocked will not be checked for input delays.

- `no_output_delay` - Reports the output ports without an output delay constraint. Output delays can be assigned using the `set_output_delay` command. Output ports that are unclocked will not be checked for output delays.

- `partial_input_delay` - Reports the input ports having partially defined input delay constraints. Assigning `set_input_delay -max` or `set_input_delay -min` to an input port, without assigning the other, creates a partially defined input delay. In such cases, paths starting from the input port may become unconstrained and no timing checks will be done against the port. Assigning `set_input_delay` without specifying either `-min` or `-max` allows the tool to assume both min and max delays, and so does not result in a partial input delay.

  Note: Unclocked input ports are not checked for partial input delays.

- `partial_output_delay` - Reports the output ports having partially defined output delay constraints. Assigning `set_output_delay -max` or `set_output_delay -min` to an output port, without assigning the other, creates a partially defined output delay. In such cases, paths reaching the port may become unconstrained and no timing checks will be done against the port. Assigning `set_output_delay` without specifying either `-min` or `-max` allows the tool to assume both min and max delays, and so does not result in a partial output delay.

  Note: Unclocked output ports are not checked for partial output delays.

- `pulse_width_clock` - Reports clock pins that have only a pulse width check associated with the pin, and no setup or hold check, no recovery, removal, or clk->Q check.

- `unconstrained_internal_endpoints` - This warning identifies timing path endpoints at register data pins that are not constrained. Endpoints at register data pins are constrained by clock assignment using the `create_clock` command. Endpoints at output ports are checked and reported by the `no_output_delay` check.

- `unexpandable_clocks` - Reports clock sets in which the period is not expandable with respect to each other, when there is at least 1 path between the clock sets. A clock is unexpandable if no common multiples are found within 1000 cycles between the source and destination clocks.

Arguments

`-file <arg>` - (Optional) Write the results to the specified file on disk. By default, the output of this command is written to the Tcl console.

Note: If the path is not specified as part of the file name, the tool will write the named file into the current working directory, or the directory from which the tool was launched.

`-append` - Append the results to the specified file. As a default the `check_timing` command will overwrite an existing file when the `-file` argument is specified.

`-no_header` - (Optional) Do not write the standard header to the report. This is a boolean option that is enabled by its use.

`-loop_limit <arg>` - (Optional) The number of loops to identify and report during the `loop` and `latch_loop` checks. The `check_timing` command will continue to perform other checks after the `-loop_limit` has been reached.

`-name <arg>` - (Optional) Creates the named report in the Timing Results view of the GUI.

`-override_defaults <args>` - (Optional) Override the default timing checks and run only the specified checks.

> 💡 **TIP:** *Multiple checks should be enclosed in quotes, "", or braces, {}.*

`-include <args>` - (Optional) Run the specified checks in addition to the current default checks.

`-exclude <args>` - (Optional) Exclude the specified checks from the default checks performed by the `check_timing` command. Specify the checks to be excluded from the list of default checks.

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-rpx <arg>` - (Optional) Specify the file name and path of an Xilinx report file (RPX) to write. This is different from writing the report results to a file using the `-file` argument. The RPX file is an interactive report that contains all the report information and can be reloaded into memory in the Vivado Design Suite using the `open_report` command. You should add a `.rpx` file extension to the specified file name, as the Vivado tool will not automatically assign a file extension.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-cells <arg>` - (Option) Perform the timing checks for the specified hierarchical cells. The details of the report will be based on the specified cells rather than the whole design.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Return more detailed results from the checks that are run. Returns details of the problems found.

Examples

The following example runs `check_timing`, but excludes the specified checks from the default timing checks:

```
check_timing -exclude {loops generated_clocks}
```

The following example uses the `-verbose` argument to obtain detailed results running just the multiple_clocks check, and then uses `get_clocks` to look further into the issue:

```
check_timing -verbose -override_defaults {multiple_clock}
  Checking multiple_clock.
  There are 2 register/latch pins with multiple clocks.
  procEngine/mode_du/set_reg[0]/C
  provEngine/mode_du/set_reg[1]/C
get_clocks -of_objects [get_pin procEngine/mode_du/set_reg[0]/C]
  sysClk coreClk
```

See Also

- create_clock
- get_clocks
- open_report
- report_timing
- set_case_analysis
- set_input_delay

Send Feedback

- set_max_delay
- set_output_delay

checkpoint_vcd

Create a VCD checkpoint (equivalent of Verilog $dumpall system task)

Syntax

```
checkpoint_vcd [-quiet] [-verbose]
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Simulation

Description

The `checkpoint_vcd` command inserts current HDL object signal values into the Value Change Dump (VCD) file. Nothing is returned. This Tcl command is the equivalent of the Verilog `$dumpall` system task, providing the initial values of the specified signals.

VCD is an ASCII file containing header information, variable definitions, and value change details of a set of HDL signals. The VCD file can be used to view simulation result in a VCD viewer or to estimate the power consumption of the design. See the *IEEE Standard for Verilog Hardware Description Language* (IEEE Std 1364-2005) for a description of the VCD file format.

You must execute the `open_vcd` and `log_vcd` commands before using the `checkpoint_vcd` command. After you execute the `checkpoint_vcd` command, run or rerun the simulation to capture the signal values.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Send Feedback

Examples

The following is an example of the `checkpoint_vcd` command where the command dumps signal values of specified HDL objects into the open VCD file:

```
checkpoint_vcd
```

See Also

- flush_vcd
- log_vcd
- open_vcd

close_bd_design

Close a design.

Syntax

```
close_bd_design [-quiet] [-verbose] <name>
```

Returns

The design object, "" if failed.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<name>	Name of design to close

Categories

IPIntegrator

Description

Closes the specified IP subsystem design in the IP integrator of the Vivado Design Suite.

If the design has been modified, you will not be prompted to save the design prior to closing. You will need to run `save_bd_design` to save any changes made to the design before using the `close_bd_design` command.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - The name of the IP subsystem design object to close.

Send Feedback

Example

The following example closes the current IP subsystem design in the current project:

```
close_bd_design [current_bd_design]
```

See Also

- create_bd_design
- current_bd_design
- get_bd_designs
- open_bd_design
- save_bd_design

Chapter 3:

close_design

Close the current design

Syntax

```
close_design [-quiet] [-verbose]
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Project](#)

Description

Closes the currently active design. If the design has been modified, you will not be prompted to save the design prior to closing. You will need to run `save_design` or `save_design_as` to save any changes made to the design before using the `close_design` command.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example closes the current design:

```
close_design
```

Note: If multiple designs are open, you can specify the current design with the `current_design` command prior to using `close_design`.

Send Feedback

The following example sets the current design, then closes it:

```
current_design rtl_1
close_design
```

current_design sets rtl_1 as the active design, then the close_design command closes it.

See Also

- current_design

close_hw_manager

Close the hardware tool

Syntax

```
close_hw_manager [-quiet] [-verbose]
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Hardware

Description

Close the Hardware Manager tool in the Vivado Design Suite.

Opening the Hardware Manager using the `open_hw_manager` command, is the first step in programming and/or debugging your design in Xilinx FPGA hardware. For more information refer to the *Vivado Design Suite User Guide: Programming and Debugging (UG908)*.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Example

The following example closes the Hardware Manager in the Vivado Design Suite:

```
close_hw_manager
```

Send Feedback

See Also

- connect_hw_server
- open_hw_manager

close_hw_target

Close a hardware target

Syntax

```
close_hw_target [-quiet] [-verbose] [<hw_target>]
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<hw_target>]	hardware target Default: current hardware target

Categories

Hardware

Description

Close the connection to the current or specified hardware target that was previously opened using the `open_hw_target` command.

The hardware target is a system board containing a JTAG chain of one or more Xilinx devices that you can program with a bitstream file, or use to debug your design. Connections between hardware targets on the system board and the Vivado Design Suite are managed by the `hw_server` application. Refer to *Vivado Design Suite User Guide: Programming and Debugging (UG908)* for a list of supported JTAG download cables and devices.

This command returns connection messages from the hardware server, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Send Feedback

<hw_target> - (Optional) Specify the hw_target object to close the connection to. The hw_target must be specified as an object as returned by the `get_hw_targets` or `current_hw_target` commands. If no target is specified, the open connection to the `current_hw_target` will be closed.

Example

The following example closes the current hardware target:

```
close_hw_target
```

See Also

- get_hw_targets
- open_hw_target
- refresh_hw_target

close_project

Close current opened project

Syntax

```
close_project [-delete] [-quiet] [-verbose]
```

Usage

Name	Description
[-delete]	Delete the project from disk also
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Project](#)

Description

Closes the current open project.

> **TIP:** *Any user-defined Tcl variables that are in the global namespace (i.e. not in a project-specific namespace) are not reset or cleared by this command. Global variables are persistent with the invocation of Vivado and are only cleared when the Vivado Design Suite is closed. You can also use the* `unset` *command to expressly clear a specific Tcl variable.*

Arguments

`-delete` - (Optional) Delete the project data from the hard disk after closing the project.

Note: Use this argument with caution. You will not be prompted to confirm the deletion of project data.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Send Feedback

Examples

The following command closes the active project:

```
close_project
```

This example closes the current project. If you have multiple projects open, the `close_project` command applies to the current project which can be defined with the `current_project` command.

The following example sets `project_1` as the current project, and then closes the project and deletes it from the computer hard disk:

```
current_project project_1
close_project -delete
```

Note: Use the `-delete` argument with caution. You will not be prompted to confirm the deletion of project data.

See Also

- current_project

Send Feedback

close_saif

Flush SAIF toggle information to the SAIF output file and close the file

Syntax

```
close_saif [-quiet] [-verbose]
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Simulation

Description

Closes the open SAIF file.

Only one SAIF file can be open in the Vivado simulator at one time, using `open_saif`. You must close the currently opened SAIF file before opening another file.

This command returns nothing if it is successful, or an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following is an example:

```
close_saif
```

Send Feedback

See Also

- log_saif
- open_saif

close_sim

Unload the current simulation without exiting Vivado

Syntax

```
close_sim [-force] [-quiet] [-verbose]
```

Usage

Name	Description
[-force]	Forces the closing of the simulation, even if changes would be lost. Default behavior is to reject the closing with an error if changes would be lost.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Simulation

Description

Close the current Vivado simulation.

Note: This command does not support third party simulators.

Arguments

`-force` - (Optional) Force closing the simulation even if changes would be lost. Does not prompt to save changes before closing.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Send Feedback

Examples

The following example closes the current simulation, forcing the close even if changes would be lost:

```
close_sim -force
```

See Also

- current_sim
- launch_simulation

Send Feedback

close_vcd

Flush VCD information to the VCD output file and close the file

Syntax

```
close_vcd [-quiet] [-verbose]
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Simulation

Description

Closes the open Value Change Dump (VCD) file.

Only one VCD file can be open in the Vivado simulator at one time. You must close the currently opened VCD file before opening another file.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example closes the current VCD object:

```
close_vcd
```

Send Feedback

See Also

- [open_vcd](#)

close_wave_config

Closes the wave config

Syntax

```
close_wave_config [-force] [-quiet] [-verbose] [<wcfgobj>]
```

Usage

Name	Description
[-force]	Forces the closing of the wave configuration, even if changes would be lost. Default behavior is to reject the closing with an error if changes would be lost.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<wcfgobj>]	Closes and destroys the specified wave configuration object, or the current wave configuration if none specified Default: NULL

Categories

Waveform

Description

Close the current, or specified wave configuration.

In the Vivado® simulator GUI, you can work with a waveform to analyze your design and debug your code. A wave configuration object displays with top-level HDL objects, and can be further populated using commands like `add_wave` and `add_wave_divider`. A new wave configuration object can be created in the current simulation with the `create_wave_config` command.

Any changes made to a wave configuration object can be saved to a Wave Config file with the `save_wave_config` command. You can open a saved Wave Config file with the `open_wave_config` command.

Arguments

`-force` - (Optional) Forces the closing of the Wave Config file, even if changes would be lost. The default behavior is to return an error if unsaved changes would be lost.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Send Feedback

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<wcfgobj> - (Optional) Specify a Wave Config object to close. The default is to close the current Wave Config file as returned by the `current_wave_config` command.

Note: The wave configuration must be specified as an object using the `get_wave_configs` command.

Examples

The following example closes all Wave Config files associated with the current simulation:

```
close_wave_config [get_wave_configs]
```

See Also

- create_wave_config
- current_wave_config
- get_wave_configs
- open_wave_config
- open_wave_database
- save_wave_config

combine_hw_platforms

Add a hw_emu Xilinx Shell Archive to an existing hw Xilinx Shell Archive

Syntax

```
combine_hw_platforms [-hw <arg>] [-hw_emu <arg>] [-o <arg>] [-force]
    [-quiet] [-verbose]
```

Usage

Name	Description
[-hw]	A hw Xilinx Shell Archive Values: A filename with alphanumeric characters and .xsa extension.
[-hw_emu]	A hw_emu Xilinx Shell Archive Values: A filename with alphanumeric characters and .xsa extension.
[-o]	The (optional) output Xilinx Shell Archive. If not specified, the hw_emu will be added to the hw file, overwriting the original file. Values: A filename with alphanumeric characters and .xsa extension.
[-force]	Overwrite existing Xilinx Shell Archive file
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Vitis, Platform, FileIO

commit_hw_hbm

Commit the property changes of the current hardware object. Inputs can be HBM or device hardware object. At least one object is required.

Syntax

```
commit_hw_hbm [-quiet] [-verbose] <hw_objects>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hw_objects>	hardware objects

Categories

Hardware

Description

Commit the current values of properties defined on the specified HBM controller in the Vivado hardware manager to the current hardware device.

The `commit_hw_hbm` command takes the current property values defined on a hw_hbm object and commits them to the current hardware device connected to the hardware server.

When you change the property values on the hw_hbm object, they are not written to the hardware device until you use the `commit_hw_hbm` command.

This command returns nothing if successful, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_objects> - (Required) Specifies the specific HBM activity monitor objects (hw_hbms) to run.

Examples

The following example changes a property of the specified HBM controller object, and then commits the change to the device:

```
set_property MC2.INIT.AM_REPEAT_EN 1 [get_hw_hbms *HBM_2]
commit_hw_hbm [get_hw_hbm *HBM_2]
```

See Also

- add_hw_hbm_pc
- current_hw_device
- current_hw_target
- get_hw_hbms
- pause_hw_hbm_amon
- refresh_hw_hbm
- remove_hw_hbm_pc
- resume_hw_hbm_amon
- run_hw_hbm_amon
- stop_hw_hbm_amon

commit_hw_mig

Commit the property changes of the current hardware object. Inputs can be any mig, device, target, or server hardware object. At least one object is required.

Syntax

```
commit_hw_mig [-quiet] [-verbose] <hw_objects>
```

Usage

Name	Description
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<hw_objects>`	hardware objects

Categories

Hardware

Description

Commit the current values of properties defined on the specified memory IP debug core hardware objects in the Hardware Manager feature of the Vivado Design Suite to the current hardware device.

The `commit_hw_mig` command takes the current property values defined on a hw_mig object in the Vivado logic analyzer, and commits them to the current hardware device connected to the hardware server.

When you change the property values on the hw_mig object, like the CONFIG.* properties, they are not written to the hardware device until you use the `commit_hw_mig` command.

This command returns nothing if successful, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Send Feedback

Note: Message limits can be defined with the `set_msg_config` command.

<hw_objects> - (Optional) Inputs can be any hw_mig, hw_device, hw_target, or hw_server object. If no object is specified, the current hw_device is targeted.

Note: The objects must be specified using the appropriate `get_hw_xxx` command, for instance `get_hw_sysmon`, rather than specified by name.

Example

The following example commits the current properties on the hw_mig object to the current hardware device:

```
commit_hw_mig [lindex [get_hw_migs] 0]
```

See Also

- connect_hw_server
- current_hw_device
- get_hw_migs
- implement_mig_cores
- refresh_hw_mig
- report_hw_mig
- set_property

commit_hw_sio

Commit the property changes of the current hardware object. Inputs can be any serial I/O (except scan and sweep), device, target, or server hardware object. At least one object is required.

Syntax

```
commit_hw_sio [-quiet] [-verbose] <hw_objects>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hw_objects>	hardware objects

Categories

Hardware

Description

Commit the current values of properties defined on the specified serial I/O hardware objects in the Hardware Manager feature of the Vivado Design Suite to the current hardware device.

Specified objects can include any serial I/O object such as GTs, RXs, TXs, PLLs, or Commons, excluding hw_sio_scan and hw_sio_sweep objects. SIO objects also include device, target, or server hardware objects.

The SIO IBERT core operates on an object property-based set/commit use model. You can set the property values on the hardware objects using the `set_property` command. You can then drive those values onto the current hardware device using the `commit_hw_sio` command.

💡 **TIP:** *To update the properties on the hardware object with the actual value on the device use the* `refresh_hw_sio` *command.*

This command returns nothing if successful, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_objects> - (Required) Specify one or more hw_sio objects to refresh. The hw_sio objects must be specified as objects returned by one of the `get_hw_sio_*` commands.

Example

The following example sets the value of the CPLL_REFCLK_DIV property on the specified serial I/O GT object, and then commits the values of all properties on that object to the current hardware device:

```
set_property CPLL_REFCLK_DIV 3 [get_hw_sio_gts *MGT_X0Y8]
commit_hw_sio [list [get_hw_sio_gts *MGT_X0Y8}] ]
```

See Also

- current_hw_device
- get_hw_devices
- get_hw_servers
- get_hw_sio_commons
- get_hw_sio_gts
- get_hw_sio_iberts
- get_hw_sio_links
- get_hw_sio_plls
- get_hw_sio_rxs
- get_hw_sio_txs
- get_hw_targets
- report_property
- set_property

commit_hw_sysmon

Commit the property changes of the current hardware object. Inputs can be hw_server, hw_target, hw_device or hw_sysmon objects. At least one object is required.

Syntax

```
commit_hw_sysmon [-quiet] [-verbose] <hw_objects>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hw_objects>	hardware objects

Categories

Hardware

Description

The `commit_hw_sysmon` command takes the current property values defined on a hw_sysmon object, and commits them to the system monitor registers on the hardware device.

When you change the property values on the hw_sysmon object, like the CONFIG.* properties, they are not written to the hardware device until you use the `commit_hw_sysmon` command.

This command returns nothing if successful, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_objects> - (Optional) Commit the properties of the specified hw_sysmon object. The system monitor object can be specified as the hw_sysmon object, or as the system monitor through the associated hw_device, hw_target, or hw_server objects.

Note: The objects must be specified using the appropriate `get_hw_xxx` command, for instance `get_hw_sysmon`, rather than specified by name.

Example

The following example sets the unipolar/bipolar configuration register property on the hw_sysmon object, and then commits that value to the system monitor of the current hardware device:

```
set_property CONFIG_REG.BU 1 [get_hw_sysmon]
commit_hw_sysmon [lindex [get_hw_sysmons] 0]
```

See Also

- connect_hw_server
- current_hw_device
- get_hw_sysmons
- get_hw_sysmon_reg
- refresh_hw_sysmon
- set_hw_sysmon_reg
- set_property

Send Feedback

commit_hw_vio

Write hardware VIO probe OUTPUT_VALUE properties values to VIO core(s).

Syntax

```
commit_hw_vio [-quiet] [-verbose] <hw_objects>...
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hw_objects>	List of hardware VIO and hardware probe objects.

Categories

[Hardware](#)

Description

Commit the current values defined on the probes of the VIO Debug core to the current hardware device.

The Virtual Input/Output (VIO) debug core can both monitor and drive internal signals on a programmed Xilinx FPGA in real time. The VIO core uses hardware probes, hw_probe objects, to monitor and drive signals on the device. Input probes monitor signals as inputs to the VIO core. Output probes drive signals to specified values from the VIO core.

The VIO core operates on an object property-based set/commit use model. You can set the OUTPUT_VALUE property on the output probes of the VIO core using the `set_property` command. You can then drive those values onto probed signals on the hardware device using the `commit_hw_vio` command.

This command returns nothing if successful, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

Send Feedback

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_objects> - (Optional) Specify the hw_vio debug core, or the hw_probe objects, to commit the OUTPUT_VALUE properties onto the hw_device. You can commit the values of one or more hardware probes, or you can commit the values on all the probes of the VIO core by specifying the hw_vio object.

Note: The objects must be specified using the `get_hw_vios` or `get_hw_probes` commands, rather than specified by name.

Example

The following example demonstrates the OUTPUT_VALUE property of the reset hw_probe being set high and committed to the hw_device, to trigger the reset process. Then the value is set low to release the reset:

```
set_property OUTPUT_VALUE 1 [get_hw_probes fast_cnt_reset \
    -of_objects [get_hw_vios hw_vio_1]]
commit_hw_vio [get_hw_probes {fast_cnt_reset} \
    -of_objects [get_hw_vios hw_vio_1]]
set_property OUTPUT_VALUE 0 [get_hw_probes fast_cnt_reset \
    -of_objects [get_hw_vios hw_vio_1]]
commit_hw_vio [get_hw_vios hw_vio_1]
```

Note: In the first occurrence of the `commit_hw_vio` command, a single hardware probe is specified as the object, while the whole hw_vio debug core is specified in the second occurrence of the command.

See Also

- connect_hw_server
- current_hw_device
- get_hw_probes
- get_hw_vios
- program_hw_devices
- refresh_hw_vio
- reset_hw_vio_activity
- reset_hw_vio_outputs
- set_property

compile_c

Compile C code into RTL

Syntax

```
compile_c [-force] [-quiet] [-verbose] <objects>
```

Usage

Name	Description
[-force]	Force generate product state regeneration
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<objects>	The objects which need C to RTL conversion

Categories

Project, IPFlow, IPIntegrator

Description

In IP cores that are imported from Vivado HLS, the `compile_c` command detects C, C++, and SystemC files and converts those files to RTL for synthesis by the Vivado Design Suite.

This lets you use Vivado HLS to describe IP cores in a high-level language, like C or C++ rather than RTL.

When HLS-based IP cores are generated, they only deliver the C source. When the HLS-based IP is synthesized, either in the out-of-context flow, or with the top-level design, the `compile_c` command launches Vivado HLS to convert the C source files into RTL, and import the resulting RTL sources back into the design prior to synthesis.

Note: The `compile_c` command is automatically called by the Vivado Design Suite when it encounters IP with C code from the Vivado HLS system. You should not need to manually call this command.

Arguments

`-force` - (Optional) Force regeneration of the RTL in the HLS-based IP.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

Send Feedback

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<objects>` - (Required) Specify the IP objects in the current project to convert files from C, C++, or SystemC to RTL code.

Example

The following example gets the C-language files from the specified IP object and converts them to RTL:

```
compile_c [get_ips instance_name]
```

See Also

- get_ips

compile_simlib

Compile simulation libraries

Syntax

```
compile_simlib [-directory <arg>] [-family <arg>] [-force]
    [-language <arg>] [-library <arg>] [-print_library_info <arg>]
    -simulator <arg> [-simulator_exec_path <arg>] [-gcc_exec_path <arg>]
    [-source_library_path <arg>] [-no_ip_compile] [-no_systemc_compile]
    [-32bit] [-quiet] [-verbose]
```

Usage

Name	Description
[-directory]	Directory path for saving the compiled results Default: .
[-family]	Select device architecture Default: all
[-force]	Overwrite the pre-compiled libraries
[-language]	Compile libraries for this language Default: all
[-library]	Select library to compile Default: all
[-print_library_info]	Print Pre-Compiled library information
-simulator	Compile libraries for this simulator
[-simulator_exec_path]	Use simulator executables from this directory
[-gcc_exec_path]	Use GNU executables from this directory
[-source_library_path]	If specified, this directory will be searched for the library source files before searching the default path(s) found in environment variable XILINX_VIVADO for Vivado
[-no_ip_compile]	Do not compile IP static files from repository
[-no_systemc_compile]	Do not compile models for system simulation
[-32bit]	Perform the 32-bit compilation
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Simulation

Description

Compile Xilinx® simulation libraries for the cells and IP used in the current project, or from a specified directory for use in multiple design projects.

Send Feedback

The Vivado Design Suite provides simulation models as a set of files and libraries that contain the behavioral and timing models for use by the Vivado simulator. The `compile_simlib` command compiles these libraries for use by third-party simulators prior to design simulation. Libraries must generally be compiled or recompiled with a new software release to update simulation models and to support a new version of a simulator.

> **IMPORTANT!** *You should rerun the* `compile_simlib` *command any time a new third party simulator will be used, or a new Vivado Design Suite version or update is installed.*

When this command is run from a current project, the tool will use the device family, target language, and library settings specified by the project as the default values, rather than the default settings of the command defined below. The default settings can be overridden by specifying the necessary options when the command is run.

The `compile_simlib` command uses simulator compilation directives when compiling the simulation libraries. You can edit the default configuration settings using the `config_compile_simlib` command.

The command returns information related to the compiled libraries, or an error if it fails.

Arguments

`-directory <arg>` - (Optional) Directory path for saving the compiled library results.

Note: By default, the libraries are saved in the current working directory in Non-Project mode, and the libraries are saved in `<project>/<project>.cache/compile_simlib` directory in Project mode. Refer to the *Vivado Design Suite User Guide: Design Flows Overview (UG892)* for more information on Project and Non-Project modes.

`-family <arg>` - (Optional) Compile simulation libraries for the specified device family. If this switch is not specified, the libraries for all families will be compiled.

<arg>: all, kintex7, virtex7, artix7, spartan7, zynq, kintexu, kintexuplus, virtexu, virtexuplus, zynquplus, zynquplusrfsoc, versal

- all (generate libraries for all device families, the default)
- kintex7 - Kintex-7 family
- virtex7 - Virtex-7 family
- artix7 - Artix®-7 family
- spartan7 - Spartan-7 family
- zynq - Zynq-7000 family
- kintexu - Kintex UltraScale family
- kintexuplus - Kintex® UltraScale+ family

Send Feedback

- virtexu - Virtex UltraScale™ family

- virtexuplus - Virtex® UltraScale+™ family

- zynquplus - Zynq® UltraScale+ family

- zynquplusrfsoc - Zynq® UltraScale+ RFSoCs family

- versal - Versal™ ACAP family

`-force` - (Optional) Overwrite the current pre-compiled libraries.

`-language [ verilog | vhdl | all ]` - (Optional) This option is only needed for use with `-no_ip_compile`, and will compile base simulation libraries for the specified language. If this option is not specified then the language will be set according to the simulator selected with `-simulator`. For multi-language simulators both Verilog and VHDL libraries will be compiled.

> 💡 **TIP:** *By default, `compile_simlib` compiles simulation libraries for IP, and compiles all languages for the IP.*

`-library <arg>` - (Optional) Specify the simulation library to compile. As a default, the `compile_simlib` command will compile all simulation libraries. Valid values are:

- all (the default)

- unisim

- simprim

To specify multiple libraries, repeat the `-lib` options for each library. For example:

```
.. -library unisim -library simprim ..
```

`-print_library_info` - (Optional) Print the library information for the compiled simulation library.

`-simulator <arg>` - (Required) Compile simulation libraries for the specified simulator.

<arg>: modelsim, questa (or questasim), ies, xcelium, vcs (or vcs_mx), riviera, activehdl (or active_hdl)

- `modelsim` - ModelSim Simulator version 2020.4 or later

- `questa` - Questa Advanced Simulator version 2020.4 or later

- `ies` - (Linux only) Incisive Enterprise Simulator version 15.20.083 or later

- `xcelium` - (Linux only) Xcelium Parallel Simulator version 20.09.006 or later

- `vcs_mx` - (Linux only) Verilog Compiler Simulator version R-2020.12 or later

- `riviera` - Riviera-PRO Simulator version 2020.10 or later

Send Feedback

- `activehdl` - (Windows only) Active-HDL Simulator version 12.0

`-simulator_exec_path <arg>` - (Optional) Specify the directory to locate the third-party compiler and simulator executables. This option is required if the target simulator is not specified in the $PATH or %PATH% environment variable; or to override the path from the $PATH or %PATH% environment variable.

`-gcc_exec_path <arg>` - (Optional) Specify the directory to locate the third-party simulator GCC executables. If this switch is not specified, then the directory path will be determined from the path set by the `GCC_SIM_EXE_PATH` environment variable. If this variable is not set, then compile_simlib will automatically try to calculate the path from the simulator installation. If not able to determine, the path will be then searched from the `$PATH` (Linux) or `%PATH%` (Windows) environment variable.

Supported GCC versions.

- questa - 5.3.0

- xcelium - 6.3

- vcs - 6.2.0

`-source_library_path <arg>` - (Optional) If specified, this directory will be searched for the library source files before searching the default path(s) defined by the environment variables ($XILINX or $XILINX_VIVADO).

Note: Do not use this option unless explicitly instructed to by Xilinx Technical Support.

`-no_ip_compile` - (Optional) Disables the compilation of simulation files for IP in the design or the specified repositories. By default, the `compile_simlib` command compiles the static simulation files for all IP in the IP catalog, including added user and third-party repositories. Use this option to disable that feature.

`-no_systemc_compile` - (Optional) Disables the compilation of Systemc/C++/C based IPs. If your design does not need to be verified for system simulation then you can specify this option.

Note: By default, the `compile_simlib` command will compile these IPs.

`-32bit` - (Optional) Perform simulator compilation in 32-bit mode instead of the default 64-bit compilation.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

Compile UNISIM and SIMPRIM libraries for ModelSim (VHDL) for a design using a Virtex-7 device.

```
compile_simlib -simulator modelsim -family virtex7 -library unisim \
    -library simprim -language vhdl
```

See Also

- config_compile_simlib
- export_simulation
- launch_simulation

Send Feedback

config_compile_simlib

Configure settings for compile_simlib

Syntax

```
config_compile_simlib [-cfgopt <arg>] [-simulator <arg>] [-reset] [-quiet]
    [-verbose]
```

Usage

Name	Description
`[-cfgopt]`	Configuration option in form of simulator.language.library.options
`[-simulator]`	Display the configurations for specified simulator
`[-reset]`	Reset all configurations
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution

Categories

Simulation

Description

Configure third party simulator options for use by the `compile_simlib` command.

The Vivado Design Suite has a pre-defined configuration file for the `compile_simlib` command, with compilation options defined for supported third-party simulators, languages, and libraries. The `config_compile_simlib` command is provided to let you change the configuration options for specific combinations of simulator, language, and library.

Use the `config_compile_simlib` command without any arguments to return all current configuration options.

Arguments

`-cfgopt <arg>` - (Optional) Specify the configuration options for a specific third party simulator, language, and library combination. The `-cfgopt` argument is specified as a string made up of four parts in the form:

{<simulator>.<language>.<library>.<options> }

Where:

Send Feedback

- `<simulator>` - Specify the third-party simulator to configure options for. Refer to the `compile_simlib` command for currently supported versions of third-party simulators. Valid values are:

    ○ `modelsim`

    ○ `questasim`

    ○ `ies`

    ○ `vcs_mx`

    ○ `riviera`

    ○ `active_hdl`

- `<language>` - Specify the language to set the simulation options for. Valid values are verilog or vhdl.

- `<library>` - Specify the library to compile. Valid values for library are:

    ○ axi_bfm

    ○ ieee

    ○ simprim

    ○ std

    ○ unisim

    ○ vl

- `<options>` - Configuration options specific to the simulator, language, and library specified. The following are the default compilation options available for different `<simulator>`.`<language>`.`<library>` combinations:

    ○ Active HDL: -v2k5 (verilog), +define+XIL_TIMING, -93 (vhdl), -nowarn ELAB1_0026 (vhdl)

    ○ Incisive Enterprise Simulator: -MESSAGES, -NOLOG, -DEFINE XIL_TIMING, -v93 (vhdl), -RELAX (vhdl)

    ○ ModelSim: -novopt, -quiet, +define+XIL_TIMING, -93 (vhdl), -source (simprim, unisim)

    ○ QuestaSim: -novopt, -quiet, +define+XIL_TIMING, -93 (vhdl), -source (simprim, unisim)

    ○ Riviera: -v2k5 (verilog), +define+XIL_TIMING, -93 (vhdl), -nowarn ELAB1_0026 (vhdl)

    ○ VCS MX: -sverilog (verilog), -nc, +v2k (simprim, unisim), +define+XIL_TIMING

    Note: Refer to the third-party simulator documentation for other compilation options that may be supported.

`-simulator <arg>` - (Optional) This option acts as a filter to return only the configuration options associated with the specified simulator. Refer to the `-cfgopt` option for a list of valid simulator values.

`-reset` - (Optional) Restore all settings to the default configuration options of the Vivado Design Suite.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example configures the compilation options for the Modelsim simulator, Verilog language, and Unisim library:

```
config_compile_simlib -cfgopt {modelsim.verilog.unisim: -quiet}
```

The following example configures the compilation options for multiple simulation libraries:

```
config_compile_simlib -cfgopt {modelsim.verilog.synopsys: -quiet} \
-cfgopt {modelsim.verilog.simprim:-source +define+XIL_TIMING}
```

See Also

• compile_simlib

Send Feedback

config_design_analysis

This command configures general features of design analysis.

Syntax

```
config_design_analysis [-max_common_paths <arg>] [-quiet] [-verbose]
```

Usage

Name	Description
[-max_common_paths]	Number of paths to consider for finding common paths across phases (< 20000) Default: 1000
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Timing](#)

Description

This command configures features of the `report_design_analysis` command.

The design analysis report analyzes timing paths at various stages in the Vivado tool flow, including synthesis, optimization, placement, routing. The `-max_common_paths` option specifies how many setup timing paths to capture at each stage in the flow.

Note: This command returns nothing if successful, or returns an error if it fails.

Arguments

`-max_common_paths <arg>` - (Optional) The number of paths to consider when examining the distribution of common paths across phases of implementation. Specified as an integer value less than 20,000, the default is 1000.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Send Feedback

Examples

The following example ignores the package delays during timing analysis:

```
config_design_analysis 500
```

See Also

- report_design_analysis

Send Feedback

config_hw_sio_gts

Configure the device GTs for the specified device.

Syntax

```
config_hw_sio_gts [-dict <args>] [-quiet] [-verbose] <hw_device>
```

Usage

Name	Description
`[-dict]`	list of name/value pairs of GT settings and values to use to configure GTs
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<hw_device>`	hardware device object

Categories

[Hardware](#)

Send Feedback

config_implementation

Configure Implementation

Syntax

```
config_implementation [-quiet] [-verbose] [<list>]
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<list>]	list of config params which need to be configured

Categories

Configuration

Description

This command lets you configure implementation parameters to control the behavior of the implementation process.

The list of properties that can be configured, or have been configured by this command can be reported by the `report_config_implementation` command.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<list> - (Required) A list of parameters and their values, to configure with the command. The parameters and their values must be enclosed in braces, and the list must be enclosed in braces as show in the Example below.

Send Feedback

Examples

The following example configures implementation parameters:

```
config_implementation { {incr.ignore_user_clock_uncertainty True}
{<other_param> <other_value>} }
```

See Also

- report_config_implementation

config_ip_cache

Manage the IP instance Synthesis cache. Lists out the IP Cache entries if no options are specified.

Syntax

```
config_ip_cache [-use_cache_location <arg>] [-use_project_cache]
    [-disable_cache] [-clear_output_repo] [-clear_local_cache]
    [-cache_has_match] [-cache_was_used] [-get_id]
    [-get_entry_location <arg>] [-disk_usage_output_repo]
    [-import_from_project] [-zip_cache <arg>] [-remove] [-purge]
    [-vlnv <arg>] [-old_swvers] [-unused] [-swver <arg>]
    [-num_days_old <arg>] [-num_days_unused <arg>] [-obs_synth_crc]
    [-report] [-rptfile <arg>] [-csvfile <arg>] [-dir <arg>] [-quiet]
    [-verbose] [<ip>]
```

Usage

Name	Description
[-use_cache_location]	Set current project properties to use the specified cache location
[-use_project_cache]	Set current project properties to use the default project IP cache location
[-disable_cache]	Disable cache use.
[-clear_output_repo]	Delete from disk and in memory all cache entries that exist in the current project's designated cache (local or remote).
[-clear_local_cache]	Delete from disk and in memory all local cache entries for this project.
[-cache_has_match]	Returns the cache-ID of the cache entry that would work for this IP instance; else ''.
[-cache_was_used]	Returns '1' if the cache was used to obtain the IP's current synthesis results; else '0'.
[-get_id]	Calculate and return IP cache ID string for specified <ip>
[-get_entry_location]	Return directory where this cache-ID entry is located.
[-disk_usage_output_repo]	Return total disk usage in MB for all cache entries in the current project's ip_output_repo.
[-import_from_project]	Import existing synthesized IP from the project into the cache.
[-zip_cache]	Zip up all cache entries used by IP instances in the current project and write to the given zipfilename.
[-remove]	Remove the corresponding cache entry for the specified IP instance or specified cachedInst; return cache ID string if successful, otherwise blank.
[-purge]	Delete all cache entries that match the specified type(s): -vlnv, -obs_swvers, -obs_synth_crc, and/or -swver. Returns the number of entries deleted.
[-vlnv]	Used with -purge, specifies the VLNV of the IP the cache entries were created from. May use a wildcard ('*') in one or more fields in the VLNV.
[-old_swvers]	Used with -purge to delete cache entries created with old Vivado SW Versions.

Name	Description
`[-unused]`	Used with -purge to delete cache entries that have never been used.
`[-swver]`	Used with -purge to delete any cache entries created from this specific Vivado SW Version (i.e., '2017.1').
`[-num_days_old]`	Used with -purge to delete any cache entries that are this number of days old or older.
`[-num_days_unused]`	Used with -purge to delete any cache entries that have not been used for this number of days or longer.
`[-obs_synth_crc]`	Used with -purge to delete cache entries whose component synth checksum is not the same as the IP Catalog's current component synthesis checksum.
`[-report]`	Report cache statistics for the specified IP or cache object, or for the current cache location if none specified. If -rptfile is specified, write statistics to that file. If -dir is specified, write statistics for cache entries found under that directory.
`[-rptfile]`	Used with -report, specifies the text file to write the cache statistics to.
`[-csvfile]`	Used with -report, specifies the csv file to write the cache statistics to, in csv format.
`[-dir]`	Used eith -report, specifies the directory to report cache entry statistics for.
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`[<ip>]`	IP instance object, IP file, or IP name pattern

Categories

[Object](#), [IPFlow](#)

Description

This command lets you manage the Vivado Design Suite out-of-context (OOC) IP cache. The Vivado Design Suite caches the synthesis results for customized OOC IP in a cache repository, so multiple IP that use the same customization profile can share OOC synthesis results to decrease run time. The cached synthesis results can be reused in a single project from the project cache, or across multiple projects using a remote cache location.

When generating the synthesis output products for an OOC IP, if a matching IP customization is found in the cache repository, the cached synthesis results will be used. If no suitable match is found in the cache repository, the synthesis output products will be generated as usual and the results, including the design checkpoint (DCP), will be copied into the cache for future use.

> **TIP:** *When cached results are used, an information message is displayed in the Tcl console indicating the IP cache-ID of the results used. The cache-ID is a string value calculated from the IP configuration settings, as well as other pertinent data, such as the vivado software version, used when the cached synthesis results were first generated. This string is also used as the directory name under the cache repository where this entry's files are stored.*

Send Feedback

You can specify a location for the cached IP synthesis results either within the current project, or at an external location. The location of the IP cache is defined by the IP_OUTPUT_REPO property on a project, with the value of a string defining a valid file system directory. You can set this property using the `config_ip_cache` command with either the `-use_cache_location` or the `-use_project_cache` options. The default IP synthesis cache location is in the current project folders.

The use of the IP synthesis cache is controlled by the IP_CACHE_PERMISSIONS project property that can be set with the `set_property` command. The current values are:

- disabled - Do not use the IP synthesis cache.

- read - Use the IP synthesis cache to read OOC synthesis results from and apply as appropriate in the current project.

- write - Use the IP synthesis cache to write OOC synthesis results into, but do not use them to read IP into the current project.

- read write - Use the IP synthesis cache for both writing results to, and using those results in the current project. This is the default setting.

> 💡 **TIP:** *The IP synthesis cache can be enabled, and the cache repository specified, in the Vivado Design Suite IDE using the Settings dialog box. Refer to the Vivado Design Suite User Guide: Designing with IP (UG896) for more information on using the cache.*

Note: Cache entries created by one Vivado software version are not viable in a different Vivado software version. Local project caches are automatically cleared when updating a Vivado project to a new Vivado release.

By default, without any of the arguments specified below, the `config_ip_cache` command returns a list of entries in the IP synthesis cache, or returns an error if it fails.

Arguments

`-use_cache_location <arg>` - (Optional) Specify a directory pathname to use as the IP cache repository for the current project. For example, "`config_ip_cache -use_cache_location /mydata/global_cache`".

`-use_project_cache` - (Optional) This is the default location for the cached IP synthesis results. This option enables the current project to use the default IP cache location inside the local project directory. You can use this option to restore the default. For example, "`config_ip_cache -use_project_cache`".

`-disable_cache` - (Optional) Disable the IP synthesis cache. This has the effect of setting the IP_CACHE_PERMISSIONS property to "disabled". For example, "`config_ip_cache -disable_cache`".

> 💡 **TIP:** *To re-enable the cache after it has been disabled, you will need to set the IP_CACHE_PERMISSIONS property, or use the Vivado Design Suite IDE.*

`-clear_output_repo` - (Optional) This clears the IP synthesis cache of all existing cached results, whether the cache is local to the project or a remote directory that is shared across multiple projects. For example, "`config_ip_cache -clear_output_repo`".

⭐ **IMPORTANT!** *You will not be prompted to confirm this command before the cached results are deleted.*

`-clear_local_cache` - (Optional) This clears the IP synthesis cache of all existing cached results, but only if using the local project cache. For example, "`config_ip_cache -clear_local_cache`".

`-cache_has_match` - (Optional) Returns the cache-ID of the cached entry that would work for the specified IP instance. For example, "`config_ip_cache -cache_has_match [get_ips my_ip_0]`".

`-cache_was_used` - (Optional) Returns '1' if the cache was used to obtain the specified IP's current synthesis results; else '0'. For example, "`config_ip_cache -cache_was_used [get_ips my_ip_0]`".

`-get_id` - (Optional) Calculates and returns the cache-ID string for the specified IP instance. The cache-ID is unique to each cache entry, generated based on the IP instance configuration and current Vivado version and settings. For example, "`config_ip_cache -get_id [get_ips my_ip_0]`".

`-disk_usage_output_repo` - (Optional) Returns the total disk usage in MB for all cache entries in the current project's IP Cache repository. For example, "`config_ip_cache -disk_usage_output_repo`".

`-import_from_project` - (Optional) Populates the current IP cache with synthesis results from the current design. This can be used to populate an IP cache with the synthesis results of an existing project to be reused by other projects or design iterations. For example, `config_ip_cache -import_from_project`.

`-zip_cache <file_name>` - (Optional) Zips up all cache entries used by the IP instances in the current project and writes to the specified `<file_name>`. The cache entries could be found in the Cache repository, or in any of the loaded user repositories. If any IP instance in the project does not have a cache entry found, then the return string holds a message that lists those IP instances. For example, "`config_ip_cache -zip_cache /mydata/projCache.zip`".

`-remove` - (Optional) Removes the corresponding cache entry for the specified IP instance or specified cache object. Returns cache-ID string if successful, otherwise blank. For example, "`config_ip_cache -remove [get_ips my_ip_0]`".

`-purge` - (Optional) Deletes all cache entries that match the specified type(s): `-vlnv`, `-old_swvers`, `-unused`, `-num_days_old`, `-num_days_unused`, `-obs_synth_crc`, and/or `-swver`. Returns the number of entries deleted. More than one specified type can be used in a single `-purge` command. The specific types of entries to purge are provided by the following options:

- `-vlnv <arg>` - (Optional) Used with -purge, specifies the VLNV of the IP the cache entries were created from. Entries created from IPs with the specified VLNV will be deleted. May use a wildcard ('*') in one or more fields in the VLNV. For example, `config_ip_cache -purge -vlnv xilinx.com:ip:axi_gpio:*`.

  > 💡 **TIP:** *The VLNV is the IPDEF property on an individual IP instance.*

- `-swver <arg>` - (Optional) Used with `-purge`, deletes cache entries created by the specified software version. The software version can be specified as `<version>.<revision>`, for example `2017.2`. For example, `config_ip_cache -purge -swver 2017.2`.

- `-old_swvers` - (Optional) Used with `-purge`, deletes cache entries created by an older software version than the one currently in use. For example, `config_ip_cache -purge -old_swvers`.

- `-unused` - (Optional) Used with -purge, deletes cache entries that have never been used. For example, `config_ip_cache -purge -unused`.

- `-num_days_old <arg>` - (Optional) Used with -purge, deletes any cache entries older than the specified number of days. For example, to delete cache entries that were created 10 days or longer ago, type `config_ip_cache -purge -num_days_old 10`.

- `-num_days_unused <arg>` - (Optional) Used with -purge, deletes cache entries that have not been used for the specified number of days or longer. For example, to delete cached entries that have not been used for 10 days or longer, type `config_ip_cache -purge -num_days_unused 10`.

- `-obs_synth_crc` - (Optional) Used with -purge, deletes cache entries that were generated with the component's synthesis checksum that is different from the checksum found in the current IP catalog's component. This is only relevant if the component has been modified since the cache entry was generated. For example, `config_ip_cache -purge -obs_synth_crc`.

`-report` - (Optional) Reports cache statistics for the specified IP or cache object, or for the current cache location if none specified. Use `-rptfile` to have the report written to a text file, or `-csvfile` for a `csv` format file. Use `-dir` to report statistics for cache entries found under that directory. Cache statistics include the VLNV of the IP used to create the cached synthesis results, the number of times the cache entry was used, the date of the last time it was used, and the synthesis runtime when the entry was created. For example, to write the current cache's statistics to the return string, type `config_ip_cache -report`. To write the statistics for the cache entry that would work for the specified IP instance, to the return string, type `config_ip_cache -report [get_ips my_ip_0]`.

`-rptfile <arg>` - (Optional) Used with the `-report` option to direct the report output to the specified file name. For example, to write statistics for the current cache to a text file, type "`config_ip_cache -report /mydata/cacheStats/proj1_stats.txt`".

`-csvfile <arg>` - (Optional) Used with the -report option to direct the report output in CSV format to the specified CSV file name. For example, to write statistics for a specific IP instance's matching cache entry in CSV format to a file, type "`config_ip_cache -report -csvfile /mydata/mystats/proj1_stats.csv [get_ips my_ip_0]`".

`-dir <arg>` - (Optional) Used with the `-report` option to specify a directory (that has cache entries under it) to get the statistics for the cache entries found there. Use this to get statistics from a directory other than the current cache setting. For example, to get statistics for all the cache entries found under the specified directory to a text file, type "`config_ip_cache -report -dir /mytests/mycache -rptfile /myresults/mycacheStats.txt`".

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example returns the contents of the current IP cache, one cache entry per line:

```
join [config_ip_cache] \n
```

The following example specifies the use of an external IP cache location, and then updates the repository settings of the IP catalog:

```
config_ip_cache -use_cache_location C:/Data/ip
```

See Also

- get_ips
- set_property
- synth_design
- synth_ip

config_timing_analysis

Configure timing analysis general settings

Syntax

```
config_timing_analysis [-enable_input_delay_default_clock <arg>]
    [-enable_preset_clear_arcs <arg>] [-ignore_io_paths <arg>]
    [-disable_flight_delays <arg>] [-merge_exceptions <arg>]
    [-timing_early_launch_at_borrowing_latches <arg>]
    [-enable_time_borrowing_for_max_delay_exceptions <arg>] [-quiet]
    [-verbose]
```

Usage

Name	Description
`[-enable_input_delay_default_clock]`	Launch SDC unclocked input delays from an internally defined clock: Values: true, false; This option is not supported for UCF constraints
`[-enable_preset_clear_arcs]`	Time paths through asynchronous preset or clear timing arcs: true, false;
`[-ignore_io_paths]`	Ignore paths from primary inputs and paths to primary outputs: Values: true, false
`[-disable_flight_delays]`	Disable adding package times to IO Calculations : Values: true, false;
`[-merge_exceptions]`	Allow/Prevent the timing engine from merging timing exceptions : Values: true, false;
`[-timing_early_launch_at_borrowing_latches]`	Remove clock latency pessimism from the launching enable of paths through transparent latches. Values: auto, true, false Default: auto
`[-enable_time_borrowing_for_max_delay_exceptions]`	Allow timing paths covered by set_max_delay timing exceptions to borrow time. Values: true, false
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution

Categories

Timing

Description

This command configures general features of timing analysis.

Note: This command returns nothing if successful, or returns an error if it fails.

Arguments

`-enable_input_delay_default_clock [ true | false ]` - (Optional) Launch unclocked input delays from an internally defined clock for timing analysis. The valid values are true or false, with a default setting of false.

`-enable_preset_clear_arcs [ true | false ]` - (Optional) Enable timing paths through asynchronous preset or clear timing arcs. The valid values are true or false, with a default setting of false. By default the timing arcs of asynchronous resets are disabled in the Vivado timing engine. This option lets you enable them for timing analysis to see if problems caused by the assertion of the asynchronous reset exist in the design.

`-ignore_io_paths [ true | false ]` - (Optional) Ignore paths from primary inputs and paths to primary outputs. This lets you eliminate the net delay from the input port and to the output port as part of the timing path.

`-disable_flight_delays [ true | false ]` - (Optional) Do not add package delays to I/O calculations when this option is true.

`-timing_early_launch_at_borrowing_latches [ auto | true | false ]` - (Optional) Remove clock latency pessimism from the launching enable of paths through transparent latches. In an enabled latch the timing path for setup/hold calculations originates from the D pin of the latch instead of G pin. This option affects those timing paths. This option can be set to `true` to compensate for timing optimism when CRPR is not enabled, the Vivado timing engine uses the early launch edge from when the latch opens to calculate the launch clock latency. However, when CRPR is enabled, which is the default in the Vivado timer, this option is too conservative and should be set to `false`. The default setting is `auto` which lets the Vivado timing engine determine when to eliminate CRPR from paths through enabled latches.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example ignores the package delays during timing analysis:

```
config_timing_analysis -disable_flight_delays true
```

See Also

- config_timing_corners
- report_timing
- report_timing_summary

config_timing_corners

Configure single / multi corner timing analysis settings

Syntax

```
config_timing_corners [-corner <arg>] [-delay_type <arg>] [-setup] [-hold]
    [-quiet] [-verbose]
```

Usage

Name	Description
[-corner]	Name of the timing corner to be modified : Values: Slow, Fast
[-delay_type]	Type of path delays to be analyzed for specified timing corner: Values: none, max, min, min_max
[-setup]	Enable timing corner for setup analysis (equivalent to -delay_type max)
[-hold]	Enable timing corner for hold analysis (equivalent to -delay_type min)
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Timing](#)

Description

This command configures the Slow and Fast timing corners in the current design for single or multi-corner timing analysis. A synthesized or implemented design must be opened when running this command.

The variation in the manufacturing process of the physical device, and the voltage and temperature at which the device is operating, combine to create a timing corner. These three variables (PVT) determine the delay across the device. The fast corner represents a device operating with the smallest manufacturing process tolerances, the highest voltage, and the lowest temperature. The slow corner represents a device operating with the greatest manufacturing tolerances, the lowest voltage, and the highest temperature. By default the Vivado Design Suite performs both a setup and hold analysis for both slow and fast process corners, also known as quad analysis:

```
config_timing_corners -corner Slow -setup -hold
config_timing_corners -corner Fast -setup -hold
```

Send Feedback

The `config_timing_corners` command can be used to limit the default four corner analysis performed by the Vivado timing engine as appropriate to the design, to improve timing performance. To change or disable the default analysis for both corners, you must configure both the Fast and Slow corners:

```
config_timing_corners -corner Slow -delay_type max
config_timing_corners -corner Fast -delay_type none
```

Note: This command returns nothing if successful, or returns an error if it fails.

Arguments

`-corner [ slow | fast ]` - (Optional) Specifies the timing corner to be configured. Valid values are "slow" and "fast". If `-corner` is not specified, the `-delay_type` applies to both corners.

`-delay_type <value>` - (Optional) Specify the type of path delays to be analyzed for the specified timing corner. Valid values are "none", "max", "min" and "min_max". A `-delay_type` of "none" excludes the specified `-corner` from timing analysis.

> **TIP:** *Although* `-delay_type` *and* `-setup/-hold` *are both optional, one of these options are required to configure the specified corner.*

`-setup` - (Optional) Specifies setup analysis for the specified timing corner. This is the same as `-delay_type max`.

`-hold` - (Optional) Specifies hold analysis for the timing corner. This is the same as `-delay_type min`.

> **TIP:** *You can specify both* `-setup` *and* `-hold` *which is the same as* `-delay_type min_max`.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Send Feedback

Examples

The following example configures the Slow timing corner for both setup and hold analysis:

```
config_timing_corners -corner slow -setup -hold
config_timing_corners -corner slow -delay_type min_max
```

Note: The two preceding examples have the same effect.

The following example configures the Fast corner for min delay analysis, and disables the Slow corner analysis:

```
config_timing_corners -corner fast -delay_type min
config_timing_corners -corner slow -delay_type none
```

See Also

- config_timing_analysis
- report_timing

config_webtalk

Enable/disable WebTalk to send software, IP and device usage statistics to Xilinx. Note: WebTalk is always enabled for WebPACK users. WebTalk ignores user and install preference when a bitstream is generated using the WebPACK license. If a design is using a device contained in WebPACK and a WebPACK license is available, the WebPACK license will be used. To change this, please see answer record 34746.

Syntax

```
config_webtalk [-info] [-user <arg>] [-install <arg>] [-quiet] [-verbose]
```

Usage

Name	Description
[-info]	Show whether WebTalk is currently enabled or disabled
[-user]	Enable/disable WebTalk for the current user. Specify either 'on' to enable or 'off' to disable. Default: empty
[-install]	Enable/disable WebTalk for all users of the current installation. Specify either 'on' to enable or 'off' to disable. If you specify 'off', individual users will not be able to enable WebTalk using the -user option. You may need administrator rights to use this option. Default: empty
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

FileIO

Description

WebTalk is a secure design data collection feature of Xilinx software that helps Xilinx understand how you are using Xilinx devices, software, and Intellectual Property (IP).

This command returns the current state of the WebTalk feature for the current user and software installation. You can also enable or disable WebTalk to send software, IP and device usage statistics to Xilinx. No data is sent if you disable WebTalk, except for the use of the WebPACK license to generate a bitstream.

Participation in WebTalk is voluntary, except for the use of the WebPACK license. WebTalk data transmission is mandatory, and is always enabled for WebPACK users. WebTalk ignores user and install preference when a bitstream is generated using the WebPACK license.

Note: If a design is using a device contained in WebPACK and a WebPACK license is available, the WebPACK license will be used. To change this, please see answer record 34746.

Arguments

`-info` - (Optional) Returns information about the current Webtalk configuration. The state of WebTalk is dependent on both the user and install setting. If either of these settings is disabled, then WebTalk is disabled.

`-user <arg>` - (Optional) Enables or disables WebTalk for the current user.

`-install` - (Optional) Enables or disables WebTalk for the current software installation.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example returns the current state of the WebTalk configuration:

```
config_webtalk -info
INFO: [Coretcl-120] Webtalk has been disabled by the current user.
INFO: [Coretcl-123] Webtalk has been enabled for the current installation.
INFO: [Coretcl-110] This combination of user/install settings means that
WebTalk is currently disabled.
```

The following example enables WebTalk for the current user:

```
config_webtalk -user on
```

Send Feedback

connect_bd_intf_net

Connect intf_port and intf_pin list.

Syntax

```
connect_bd_intf_net [-intf_net <arg>] [-boundary_type <arg>] [-quiet]
    [-verbose] <object1> <object2> [<auto>]
```

Returns

TCL_OK, TCL_ERROR if failed.

Usage

Name	Description
`[-intf_net]`	The single intf_net that all objects connect to
`[-boundary_type]`	Used when source object is on a hierarchical block's interface pin. Valid values are 'upper', 'lower', or 'both'. If 'lower' boundary, searches from the lower level of hierarchy onwards. Default: both
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<object1>`	Name of intf_port or intf_pin to connect
`<object2>`	Name of intf_port or intf_pin to connect
`[<auto>]`	Automatically connect associated pins

Categories

IPIntegrator

Description

Connect the interface pins on an IP integrator cell to other interface pins, or to external interface ports. An interface is a grouping of signals that share a common function in the IP integrator subsystem design.

This command will create an interface net of the name specified by the `-intf_net` option, will connect to an existing interface net of the specified name, or will assign a name if none is specified.

Returns the connected interface net object, or returns an error.

Send Feedback

Arguments

`-intf_net <arg>` - (Optional) Specifies the name of an existing interface net, previously created by the `create_bd_intf_net` command, or a new interface net that will be created. If no name is provided, the IP integrator will automatically name the net.

`-boundary_type [ lower | upper | both ]` - (Optional) Specifies the search area for making a connection to an interface pin of an hierarchical block. The default of `both` searches the current level of the block design hierarchy, and downward, to find connecting objects. Specify `upper` to search only the current level of hierarchy, specify `lower` to search from the bottom of the hierarchy upward.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<object1> - (Required) The first interface pin or port to connect the net to.

<object2> - (Required) The second interface pin or port to connect the net to.

Example

The following example connects an interface pin on an IP integrator core to an interface port in the subsystem design:

```
connect_bd_intf_net [get_bd_intf_pins clk_wiz_1/CLK_IN1_D] \
    [get_bd_intf_ports /diff_clock_rtl]
```

See Also

- create_bd_cell
- create_bd_intf_net
- get_bd_cells
- get_bd_intf_pins
- get_bd_intf_ports

Send Feedback

connect_bd_net

Connect port and pin object list.

Syntax

```
connect_bd_net [-net <arg>] [-boundary_type <arg>] [-quiet] [-verbose]
    <objects>...
```

Returns

TCL_OK, TCL_ERROR if failed.

Usage

Name	Description
`[-net]`	The single net that all objects connect to
`[-boundary_type]`	Used when source object is on a hierarchical block's pin. Valid values are 'upper', 'lower', or 'both'. If 'lower' boundary, searches from the lower level of hierarchy onwards. Default: both
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<objects>`	The objects connect to the net

Categories

[IPIntegrator](#)

Description

Create a new net in the current IP integrator subsystem design connecting the specified list of block diagram port and pin objects, or connect an existing net to the specified pins and ports.

If the `-net` option is not specified, a new net is created connecting the listed objects. If `-net` is used, the specified net is either connected or created as needed.

Use the `get_bd_ports` and `get_bd_pins` commands to specify the port and pin objects to connect.

You can use this command to connect pins or ports at different levels of the subsystem design hierarchy. However, in this case, you cannot specify the `-net` option because the connection, when complete, will result in multiple nets rather than a single net.

The command returns the connected IP integrator subsystem design net object, or returns an error.

Send Feedback

Arguments

`-net <arg>` - (Optional) Create a single net in the current IP subsystem design.

Note: The `-net` argument is optional. When the objects being connected are not in the same level of hierarchy, the net argument should not be specified.

`-boundary_type [ lower | upper | both ]` - (Optional) Specifies the search area for making a connection to a pin of an hierarchical block. The default of `both` searches the current level of the block design hierarchy, and downward, to find connecting objects. Specify `upper` to search only the current level of hierarchy, specify `lower` to search from the bottom of the hierarchy upward.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<objects>` - Connect the specified list of port and pin objects in the current IP integrator subsystem design.

Examples

The following example connects two pins on different levels of the IP subsystem design hierarchy:

```
connect_bd_net [get_bd_pins /vidOut_1/locked] \
    [get_bd_pins /newMod1/bridge_1/fid]
```

Note: Because `/vidOut_1/locked` and `/newMod1/bridge_1/fid` are in different levels of the subsystem design hierarchy, the `-net` option is not specified. In this case, multiple nets are created for connection across the hierarchy.

See Also

- connect_bd_intf_net
- create_bd_net
- disconnect_bd_net
- get_bd_pins
- get_bd_ports

connect_debug_cores

Connect debug slave instances to the master instance. A valid master is a debug bridge or debug hub instance configured in "From BSCAN To DebugHUB" mode. A valid slave could be any of the following debug cores (Ex: ILA, VIO, JTAG_to_AXI). connect_debug_cores can only connect master and slave instances that exist in the same region (either in Reconfigurable Partition or static)

Syntax

```
connect_debug_cores -master <args> -slaves <args> [-quiet] [-verbose]
```

Returns

Debug master and slave instances.

Usage

Name	Description
-master	A valid debug bridge or debug hub instance configured in "From BSCAN To DebugHUB" mode. Only one master instance is allowed.
-slaves	List of valid slave instances. A valid slave instance is any of the following debug cores (Ex: ILA, VIO, JTAG_to_AXI)
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Debug, XDC

Description

Connect debug slave instances to the specified master instance. The command can add the specified slaves into an existing debug chain, where the specified slaves will be connected to the debug hub or bridge, without affecting debug slaves that are already in the connection chain.

Debug masters include both the Debug Hub and Debug Bridge. The Vivado Debug Hub core provides an interface between the JTAG Boundary Scan (BSCAN) interface of the Xilinx device and the Vivado Debug cores, including the Integrated Logic Analyzer (ILA), Virtual Input/Output (VIO), and the JTAG-to-AXI. The Vivado Debug Bridge is a debug controller that provides multiple options to communicate with the debug cores in both flat designs, or Partial Reconfiguration (PR) designs. The Debug Bridge can be configured to debug designs using a JTAG cable, or remotely through Ethernet, PCIe, or other interfaces using a Xilinx Virtual Cable (XVC), without the need for a JTAG cable. Refer to the *Vivado Design Suite User Guide: Vivado Programming and Debugging* (UG908) for more information.

> **IMPORTANT!** *For Partial Reconfiguration (PR) designs, the* `connect_debug_cores` *command can only connect master and slave instances that occur in the Static Region, or in the same Reconfigurable Partition.*

Arguments

`-master <arg>` - (Required) A valid debug hub instance, or debug bridge, configured in **From BSCAN To Debug HUB** mode for PR designs. Only one master can be specified.

`-slaves` <args> - (Required) A list of one or more debug core slave instances. A valid slave instance is an ILA, VIO, or JTAG_to_AXI debug core.

Note: If any of the specified slaves is already connected to another master, it is first disconnected from the current master and reconnected to the new master.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example connects the specified ILA cores to the debug bridge:

```
connect_debug_cores -master [get_cells inst_count/debug_bridge_0] \
-slaves [list [get_cells inst_count/ila_0] [get_cells inst_count/ila_1] ]
```

See Also

- create_debug_core
- get_cells
- get_debug_cores

- get_hw_axis
- get_hw_ilas
- get_hw_vios

Send Feedback

connect_debug_port

Connect nets and pins to debug port channels

Syntax

```
connect_debug_port [-channel_start_index <arg>] [-quiet] [-verbose] <port>
    <nets>...
```

Usage

Name	Description
[-channel_start_index]	Connect nets starting at channel index
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<port>	Debug port name
<nets>	List of nets or pins

Categories

[Debug](), [XDC]()

Description

Connects a signal from the netlist design to a port on an ILA debug core that was added to the design using the `create_debug_core` command. The signal can either be connected to a specific channel index on the port, or simply connected to an available channel on the port.

If you try to connect too many signals to a port, or there are not enough channels to support the connection, the tool will return an error.

Additional ports can be added to a debug core through the use of the `create_debug_port` command, and you can increase the available channels on an existing port with the `set_property port_width` command. See the examples below.

You can disconnect signals from ports using the `disconnect_debug_port` command.

When the debug core has been defined and connected, you can implement the debug core as a block for inclusion in the netlist design. Use the `implement_debug_core` command to implement the core.

Send Feedback

Arguments

`-channel_start_index <arg>` - (Optional) The channel index to use for the connection. If more than one signal has been specified, this is the channel index where connections will start to be added. Channel indexes are numbered starting at 0.

Note: If this argument is not specified, the tool will place connections on the first available channel index.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<port>` - (Required) The name of the port to connect signals to. The port must be referenced by the core_name/port_name.

`<nets>` - (Required) A list of one or more net names from the netlist design to connect to the specified debug port.

Examples

The following example creates a new PROBE port on the myCore debug core, increases the PORT_WIDTH property of the port in order to prepare it to receive the number of signals to be connected, and connects signals to the port starting at the third channel position (index 2).

```
create_debug_port myCore PROBE
set_property PORT_WIDTH 8 [get_debug_ports myCore/PROBE1]
connect_debug_port myCore/PROBE1 [get_nets [list m0_ack_o m0_cyc_i \
    m0_err_o m0_rty_o m0_stb_i m0_we_i ]] -channel_start_index 2
```

Note: If you specify too many nets to connect to the available channels on the port, the tool will return an error and will not connect the ports.

See Also

- create_debug_core
- create_debug_port
- disconnect_debug_port
- get_debug_ports
- get_nets
- implement_debug_core
- set_property

connect_hw_server

Open a connection to a hardware server

Syntax

```
connect_hw_server [-url <arg>] [-cs_url <arg>] [-quiet] [-verbose]
```

Returns

Hardware server

Usage

Name	Description
[-url]	hw_server url Default: localhost:3121
[-cs_url]	Url to use for cs_server. If default url is used and no server is running there, cs_server will be automatically launched on the default url. Default: TCP:localhost:3042
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Hardware

Description

⭐ **IMPORTANT!** *You must use the* `open_hw` *command to open the Hardware Manager in the Vivado Design Suite before using this command.*

To open a hardware target containing a JTAG chain of one or more Xilinx devices, for programming and/or debugging your design, you must first connect to a Vivado tools hardware server (hw_server) to manage the connection to the hardware target (hw_target).

The hw_server manages the connection to the physical programming target. It should be running on the machine connected to the hardware programmer, or test board, connected either locally or remotely. The `hw_server` command must be launched as a separate application, and can be found in the `/bin` folder of your Vivado Design Suite installation directory.

To connect to a hardware server, the `hw_server` application must be running, and the host name and port number noted for the `-url` argument of the `connect_hw_server` command. The default URL for the hw_server process is `localhost:3121`. For more information on setting up and running the Vivado hardware server, refer to the *Vivado Design Suite User Guide: Programming and Debugging (UG908)*.

You can connect a single instance of the Vivado Design Suite to multiple hardware servers, to support programming and debugging different device configurations. However, you can only have one connection to a specific hardware server as identified by the host name/port number combination. An error is returned if you attempt to open a connection to a server that is already connected.

The last connected hardware server is the current hardware server, unless changed by the `current_hw_server` command. Any connected server can be disconnected with the `disconnect_hw_server` command.

This command returns the host name of the hardware server that has been connected.

Arguments

`-url <arg>` - (Optional) The URL of the running `hw_server` application. The URL consists of <hostname>:<port_number>. The default setting is `localhost:3121`.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Example

The following example connects to the hw_server on the remote host trumpet3, through port 3121:

```
connect_hw_server -url trumpet3:3121
```

This example connects to a running hw_server that is running locally with the default url:

```
connect_hw_server
```

Note: The `hw_server` command must be run separately prior to connection.

See Also

- current_hw_server
- disconnect_hw_server
- get_hw_servers
- get_hw_targets

- [refresh_hw_server](#)

Send Feedback

connect_net

Connect a net to pins or ports

Syntax

```
connect_net [-hierarchical] [-basename <arg>] [-net <args>]
    [-objects <args>] [-net_object_list <args>] [-dict <args>] [-quiet]
    [-verbose]
```

Usage

Name	Description
[-hierarchical]	Allow hierarchical connection, creating nets and pins as needed (see -basename).
[-basename]	base name to use for net / pin names needed when doing hierarchical connection (see -hier). Default value is inferred from the name of the net being connected (see -net).
[-net]	Net to connect to given objects.
[-objects]	List of pin and port objects to connect
[-net_object_list]	optional, a list of net and pin/port list pairs, each pin or port list element is connected to the corresponding net, e.g. { net_a { pin_b port_c } net_d pin_e }. Cannot be used with -net, -objects list is ignored when -net_object_list is used.
[-dict]	alternative to -net_object_list, faster, but requires a list of net and pin/port object pairs (must be a list of objects, not names or other TCL objects), each pin or port list element is connected to the corresponding net, e.g. { $net_1 $pin_1 $net_2 $pin_2 }. Cannot be used with -net, -objects list is ignored when -dict is used.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Netlist

Description

This command allows the user to connect a specified net to one or more pins or ports in the netlist of an open Synthesized or Implemented Design.

The `connect_net` command will also connect nets across levels of hierarchy in the design, by adding pins and hierarchical nets as needed to complete the connection. Added nets and pins can be assigned a custom basename to make them easy to identify, or will be assigned a basename by the Vivado tool.

Send Feedback

> 💡 **TIP:** *You can specify multiple nets, and a list of pins and ports to connect those nets to, using a single* `connect_net` *command with the* `-net_object_list` *or* `-dict` *options, to significantly speed the addition of multiple nets to the current design.*

Netlist editing changes the in-memory view of the netlist in the current design. It does not change the files in the source fileset, or change the persistent design on the disk. Changes made to the netlist may be saved to a design checkpoint using the `write_checkpoint` command, or may be exported to a netlist file such as Verilog, VHDL, or EDIF, using the appropriate `write_*` command.

Note: Netlist editing is not allowed on the elaborated RTL design.

Arguments

`-hierarchical` - (Optional) Connect the net across different levels of the hierarchy.

> ⭐ **IMPORTANT!** *If* `-hierarchical` *is not specified, attempting to connect to hierarchical pins will fail with a warning.*

`-basename <arg>` - (Optional) Specifies a custom name to use for any hierarchical nets or pins that are needed to connect the specified net across levels of the hierarchy. If this option is not used, the basename is automatically derived from the net being connected.

`-net <arg>` - (Required) Specifies the net to connect.

> 💡 **TIP:** *Although you can create a bus using the* `-from` *and* `-to` *arguments of the* `create_net` *command, you must connect each bit of the bus separately using the* `connect_net` *command.*

`-objects <args>` - (Required) Specified the list of pins or ports to connect the net to. You can connect a net to one or more pin or port objects.

`-net_object_list <args>` - (Optional) A list of multiple nets and the pins and ports to connect those nets to. This option lets you connect multiple nets with a single `connect_net` command. Nets, pins, and ports, can be specified by name, or as objects as returned by `get_nets`, `get_pins`, or `get_ports` commands. The nets and pins/ports to connect it to are listed in the following form: `{net1 {pin1 pin2...pinN} net2 {pin1 pin2} ...netN {pin1 pin2...pinN}}`.

> 💡 **TIP:** *When* `-net_object_list` *or* `-dict` *is specified,* `-net` *and* `-objects` *should not be specified, and will be ignored by the tool. Although* `-net/-objects` *and* `-net_object_list`, *or* `-dict` *are all listed as optional, you must specify the net and objects to connect using one of those argument forms.*

`-dict` <args> - (Optional) This provides an alternate syntax to the `-net_object_list` option which offers some performance advantage. Specified as a list of net/pin or net/port object pairs that must be specified as objects, and not by name. For example:

```
set myNetA [get_nets netA]
set myPin1 [get_pins pin1]
set myPin2 [get_pins pin2]
set myPin3 [get_pins pin3]
connect_net -dict { $myNetA $myPin1 $myNetA $myPin2 $myNetA $myPin3 }
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Example

The following example creates a port; creates a pin on the myDMA instance; creates a net called myEnable; and connects the net to the newly created port and pin:

```
create_port -direction IN enableIn
create_pin -direction IN myDMA/en
create_net myEnable
connect_net -net myEnable -objects {enableIn myDMA/en}
```

The following example creates 32-bit bus ports, pins, and nets, then connects them together one bit at a time using a for loop to connect each bit individually:

```
create_port -from 0 -to 31 -direction IN dataIN
create_pin -from 0 -to 31 -direction IN myDMA/data
create_net -from 0 -to 31 dataBus
for {set x 0} {$x<32} {incr x} { \
   connect_net -net dataBus[$x] -objects [list dataIN[$x] myDMA/data[$x]] }
```

Note: Attempting to connect the dataBus will result in a "Net not found error." Each bit of the bus must be separately connected.

This example creates a new cells, then uses the `-net_object_list` to connect multiple nets in a single `connect_net` command:

```
create_cell -ref inv a2_i
connect_net -net_object_list {top_I[2] {I[2] a2_i/I} \
   top_O[2] {O[2] a2_i/O} top_clk a2_i/clk}
```

See Also

- create_net
- create_pin
- create_port
- disconnect_net
- remove_net
- resize_net_bus
- write_checkpoint
- write_edif
- write_verilog
- write_vhdl

convert_ips

Convert specified IP to or from core container format.

Syntax

```
convert_ips [-force] [-to_core_container] [-from_core_container] [-quiet]
    [-verbose] <objects>
```

Usage

Name	Description
[-force]	Force conversion even if the IP is locked.
[-to_core_container]	Convert IP to core container format.
[-from_core_container]	Convert IP to non core container format.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<objects>	Input objects for the IP. May IP or source file objects

Categories

IPFlow

Description

This command converts existing IP into core container format, or reverts core container IP into the expanded non-core container format.

The core container format for IP is a compressed zip file that reduces the file structure in the design, and increases tool performance.

By default, the Vivado tool adds IP from the Xilinx IP catalog into a design using the core container format. However, the `convert_ips` command lets you convert IP in existing designs to take advantage of the core container format. In addition, the `convert_ips` command also lets you revert the compressed core container format into the expanded non-core container format.

> **TIP:** *If neither `-to_core_container` or `-from_core_container` options are specified then the `convert_ips` command will convert the IP from its current format into the opposite form. Any core container IP will be converted to non-core container format, and any non-core container IP will be converted to core container format.*

Send Feedback

IP that is user-managed, cannot be converted from its current format. IP that is locked requires the use of the `-force` option to convert. Refer to the *Vivado Design Suite User Guide: Designing with IP (UG896)* for more information on editing IP and the IS_LOCKED and IS_MANAGED properties.

This command returns a transcript of its actions, or returns an error if it fails.

Arguments

`-force` - (Optional) Force the conversion of IP that are currently locked (IS_LOCKED).

`-to_core_container` - (Optional) Convert existing expanded form IP into the core container format. Any IP specified that are already in core container format will simply be ignored.

`-from_core_container` - (Optional) Convert IP currently in the core container format into the expanded form of the non-core container format. Any IP specified that are already in non-core container format will simply be ignored.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<objects>` - (Required) Specify the IP objects or IP files to convert. IP files can be specified using the `get_files` command, or IP objects can be specified with `get_ips`.

Examples

The following example converts all IP in the current project into core container format:

```
convert_ips -to_core_container [get_ips]
```

Note: Any IP already in the core container format will be skipped.

The following example converts the specified IP file to core container format:

```
convert_ips -to_core_container \
[get_files  C:/Data/wave1/wave1.srcs/sources_1/ip/char_fifo/char_fifo.xci]
```

The following example toggles the current format of all IP in the design, switching from core container to non-core container, and from non-core container to core container:

```
convert_ips [get_ips]
```

See Also

- get_files
- get_ips
- get_property
- set_property

convert_ngc

(User-written application) Convert all provided NGC files to a supported format

Syntax

```
convert_ngc [-output_dir <arg>] [-format <arg>] [-add_to_project] [-force]
    [-quiet] [-verbose] <files>
```

Returns

None

Usage

Name	Description
[-output_dir]	Directory to place all output, else the output is placed at location of NGC file Default: Script output directory path
[-format]	Accepts 'Verilog' or 'EDIF' (Default: EDIF), specifies the desired output format Default: EDIF
[-add_to_project]	Adds the output files to the current project, if no project is open, then this option does nothing
[-force]	Force overwriting of files that already exist on disk, replaces files in project if add_to_project switch was specified
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<files>	A list of NGC files to convert

Categories

[xilinxtclstore](), [projutils]()

Description

Converts provided NGC files to a supported format.

Arguments

`-output_dir <arg>` - (Optional) Directory to place all output. If not specified, the output is placed in the same directory as the input NGC file.

`-format [ Verilog | EDIF ]` - (Optional) Specifies the desired output format. The default is EDIF.

`-add_to_project` - (Optional) Adds the output files to the current project, if no project is open, then this option is ignored.

Send Feedback

`-force` - (Optional) Force overwriting of files that already exist on disk, replaces files of the same name in the current project, if the `-add_to_project` option is specified.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<files>` - (Required) A list of NGC files to convert.

Examples

The following example will convert `test.ngc` to `test.edn`, with verbose messaging. The `test.edn` file will be added to the current open project:

```
convert_ngc ./test.ngc -add_to_project -verbose
```

The following example will convert `test.ngc` to `test.edn`. The `test.edn` file will be placed in the `./output` directory. If `./output/test.edn` exists it will be replaced:

```
convert_ngc ./test.ngc -output_dir output -force
```

The following example converts all NGC files in the current directory and in all sub-directories:

```
convert_ngc [ glob ./**/*.ngc ] [ glob ./*.ngc ]
```

See Also

- add_files
- current_project
- get_files
- read_edif

copy_bd_objs

Make copies of the objects and add the copies to the given hierarchical cell.

Syntax

```
copy_bd_objs [-prefix <arg>] [-from_design <arg>] [-quiet] [-verbose]
    <parent_cell> <objects>...
```

Returns

0, "" if failed.

Usage

Name	Description
`[-prefix]`	Prefix name to add to cells
`[-from_design]`	The design to own the original objects
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<parent_cell>`	Parent cell
`<objects>`	The objects to copy

Categories

[IPIntegrator](#)

Description

Use this command to copy IP integrator objects from one open subsystem design to a second subsystem design. The selected objects can be copied into the top-level of the current subsystem design, or into an existing hierarchical module.

Because `get_bd_cells`, and other commands like it, operate on the current subsystem design, you must store the objects to be copied in a Tcl variable, as shown in the example below. Set the current subsystem design to the source design, select the group of objects to be copied, and store them in a Tcl variable. Then change the `current_bd_design` to the target design, and copy the selected objects. In this case, the `-from_design` option must be used.

You can also use this command to copy objects from one level of hierarchy in the current subsystem design to another. In this case, the `-from_design` argument does not need to be specified.

This command returns 0 if successful, and returns an error if it failed.

Send Feedback

Arguments

`-prefix <arg>` - (Optional) A prefix name to apply to any cells that are copied into the hierarchical module.

`-from_design <arg>` - (Optional) The name of the IP integrator subsystem design where the specified objects are located. The design must be open in IP integrator. If `-from_design` is not specified, the objects are located in the current design, as determined by the `current_bd_design` command.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<parent_cell>` - (Required) The name of the hierarchical module to copy the specified objects into. You can specify "/" for the top-level of the current subsystem design.

`<objects>` - (Required) The list of cells and nets specified by the `get_bd_cells` and `get_bd_nets` commands, to copy into the specified `<parent_cell>`.

Example

The following example sets the current subsystem design, selects a group of objects, and stores them in a Tcl variable. The current subsystem design is then changed, and the selected objects are copied into the top-level of the current design:

```
current_bd_design myDesign
myDesign
set copyObjs [get_bd_cells {vidOut_1 bridge_1}]
/vidOut_1 /bridge_1
current_bd_design design_1
design_1
copy_bd_objs -from_design myDesign / $copyObjs
0
```

Note: Because the `get_bd_cells` command only returns cells from the current subsystem design, the Tcl variable is used to store those objects prior to switching to the target design with `current_bd_design`.

See Also

- current_bd_design
- get_bd_cells

Send Feedback

- get_bd_nets

Send Feedback

copy_constraints

creates a copy of given constraint set

Syntax

```
copy_constraints [-constrset <arg>] [-quiet] [-verbose] <name>
```

Returns

Returns the copied constraint set object.

Usage

Name	Description
[-constrset]	Contraint FileSet from which files are copied
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<name>	Name of the copied constraint set

Categories

Project

Send Feedback

copy_ip

Copy an existing IP

Syntax

```
copy_ip -name <arg> [-dir <arg>] [-quiet] [-verbose] <objects>...
```

Returns

IP file object that was added to the project.

Usage

Name	Description
-name	Name of copied IP
[-dir]	Directory path for remote IP to be created and managed outside the project
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<objects>	IP to be copied

Categories

Project, IPFlow

Description

Create a copy of an IP core that has been previously instanced into the current project.

Arguments

`-name <arg>` - (Required) Specify the name of the new IP to be created.

`-dir <arg>` - (Optional) The path to a directory outside of the local project to store the newly created IP. The specified directory must already exist, or the command will return an error.

Note: If a directory is not specified, the new IP will be added to the local project directory structure at `<project_name>.srcs/sources_1/ip`.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

Send Feedback

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<object>` - (Required) The IP object to copy. The `copy_ip` command can only be used to copy a single IP core at one time. The IP must be specified as an IP object returned by the `get_ips` command, and not simply referenced by name.

Example

The following example create a copy of the FIFO core previously instanced into the current project and writes it to the specified directory:

```
copy_ip -name newFIFO -dir C:/Data/new_IP [get_ips oldFIFO]
```

See Also

- create_ip
- get_ips
- import_ip
- read_ip

copy_run

(User-written application) Copy a run from an already existing run, source-run, to a new copy of that run, destination-run.

Syntax

```
copy_run [-parent_run <arg>] [-verbose] -name <arg> [-quiet] <run>
```

Returns

The new run object.

Usage

Name	Description
[-parent_run]	Specify the synthesis run for the new implementation run, accepts name or run object (Default: same as source run) Default: None
[-verbose]	Print detailed information as the copy progresses
-name	Specify the name of the new run
[-quiet]	Ignore command errors
<run>	The run to be copied, accepts name or run object

Categories

xilinxtclstore, projutils

Description

Copies an existing synthesis or implementation run.

Arguments

`-name <arg>` - (Optional) Specify the name for the new run.

`-parent_run <arg>` - (Optional) Specify the synthesis run for a new implementation run. Can be specified as the run name, or the run object as returned by `get_runs` or `current_run`.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<run>` - (Required) The run to be copied. Can be specified as the run name, or the run object as returned by `get_runs` or `current_run`.

Examples

The following example will copy synth_1 run into a new synth_2 run:

```
copy_run -name synth_2 [get_runs synth_1]
```

The following example will copy the impl_1 run into a new run called impl_2, and assign synth_2 as the parent of the new run:

```
copy_run -name impl_2 [get_runs impl_1] -parent_run synth_2
```

See Also

- create_run
- current_run
- get_runs

create_bd_addr_seg

Create a new segment.

Syntax

```
create_bd_addr_seg -range <arg> -offset <arg> [-quiet] [-verbose]
    [<parent_addr_space>] [<slave_segment>] <name>
```

Returns

The newly created segment object, "" if failed.

Usage

Name	Description
`-range`	Range of segment. e.g. 4096, 4K, 16M, 1G
`-offset`	Offset of segment. e.g. 0x00000000
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`[<parent_addr_space>]`	Parent address space of segment
`[<slave_segment>]`	Slave segment of the created segment
`<name>`	Name of segment to create

Categories

IPIntegrator

Description

Create a new address segment object, bd_addr_seg, in the current IP integrator subsystem design.

This command returns the newly created master address segment object, or returns nothing if it failed.

Arguments

`-range <arg>` - (Required) Range, or size, of address segment to create expressed as an integer or hexadecimal value. The range should be specified as a number of bits expressed as a power of 2, such as 4096, or as an amount of memory to allocate such as 4K, 16M, 1G.

`-offset <arg>` - (Required) Offset of the address segment. Written as an integer or hexadecimal value, such as 0x00000000.

Send Feedback

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<parent_addr_space> - (Required) Defines the parent address space of the segment.

`<slave_segment>` - (Required) Maps a slave address segment of the master address segment being created.

`<name>` - (Required) The name of the master address segment to create.

Example

The following example creates address segments for the Data and Instruction address spaces of the Microblaze core:

```
create_bd_addr_seg -range 0x10000 -offset 0x41200000 \
    [get_bd_addr_spaces microblaze_1/Data] \
    [get_bd_addr_segs microblaze_1_axi_intc/s_axi/Reg] SEG1
create_bd_addr_seg -range 0x40000000 -offset 0x0 \
    [get_bd_addr_spaces microblaze_1/Instruction] \
    [get_bd_addr_segs microblaze_1_local_memory/ilmb_bram_if_cntlr/SLMB/Mem]
\
    SEG1
```

See Also

- assign_bd_address
- exclude_bd_addr_seg
- get_bd_addr_segs
- get_bd_addr_spaces
- include_bd_addr_seg

create_bd_cell

Add an IP cell from the IP catalog, or add a new hierarchical block.

Syntax

```
create_bd_cell [-vlnv <arg>] [-type <arg>] [-reference <arg>]
    [-revision <arg>] [-quiet] [-verbose] <name>
```

Returns

The newly created cell object. Returns nothing if the command fails.

Usage

Name	Description
[-vlnv]	Vendor:Library:Name:Version of the IP cell to add from the IP catalog.
[-type]	Type of cell to create. Valid values are IP, hier, container and module. Default: IP
[-reference]	Top module-name or file-path of the module which is referred to create the cell.
[-revision]	(Optional) Core revision Default: -1
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<name>	Name of cell to create

Categories

IPIntegrator

Description

Add a cell from the Vivado catalog to the current subsystem design, create a new hierarchical module to add to the subsystem design, or create a new module by referencing the module definition from an HDL source file.

When adding an IP core from the catalog, the `-vlnv` argument is required.

When creating a new hierarchical block design module, the `-type hier` argument is required.

Send Feedback

When creating a block design module that references an RTL module or entity declaration the `-type module` argument is required, as well as `-reference`. The module reference feature lets you add a module definition from an RTL file (Verilog or VHDL) into the block design. The source file containing the module definition must be added to the project, or read into the design before creating a module reference. Refer to the *Vivado Design Suite User Guide: Designing IP Subsystems Using IP Integrator* (UG994) for more information on referencing modules.

This command returns the name of the newly created cell object, or returns nothing if the command fails.

Arguments

`-vlnv <arg>` - (Required) This option is required for `-type IP` (the default), or optional for `-type hier`. Specify the Vendor:Library:Name:Version attribute of the cell to add from the IP integrator catalog. The VLNV attribute identifies the object in the IP integrator catalog. This argument is not needed when creating a new hierarchical module.

Note: The `-vlnv` property for IP from the IP integrator catalog refers to files in the Vivado Design Suite installation hierarchy that can be found at `data/ip/xilinx`.

`-type [ IP | hier | container | module ]` - (Optional) Specify the cell as being:

- `IP`: IP from the Vivado IP catalog. This is the default type of block design cell created, but requires the use of the `-vlnv` option.

- `hier`: A new empty hierarchical block design module to add to, and populate in the current design.

- `container`: A block design container referenced from a BD file loaded in the source fileset.

  This requires the use of the `-reference` option.

- `module`: A hierarchical module referenced from a Verilog or VHDL file loaded in the source fileset. This requires the use of the `-reference` option.

Note: Although both `-vlnv` and `-type` are marked as Optional, one or the other must be specified. Use `-vlnv` to identify the IP core to add from the Vivado IP catalog, or specify `-type` to create or reference a hierarchical module.

`-reference <arg>` - (Optional) Specifies the module name to reference from a loaded RTL design source file.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - The name of the IP cell or hierarchical module to add to the current IP subsystem design.

Examples

This example adds an AXI FIFO core from the IP integrator catalog to the current subsystem design, with the specified name:

```
create_bd_cell -vlnv xilinx.com:ip:axi_fifo_mm_s:4.0 axi_fifo_1
```

Note: The `-vlnv` argument identifies the core to add from the Vivado catalog.

This example creates a new block design container in the block design, referencing the specified module definition from a previously loaded BD source file:

```
set bdc [ create_bd_cell -type container -reference hier_0 bdc ]
```

This example creates a new module in the block design, referencing the specified module definition from a previously loaded RTL source file:

```
create_bd_cell -type module -reference rtlRam rtlRam_0
```

This example creates a new hierarchical module, myModule1, and moves the AXI FIFO from the prior example into the new module. myModule1 is set as the current instance in the subsystem design, and a new module is created, myModule2, which is added to the current instance. Finally the current instance is reset to point to the top-level of the subsystem design:

```
create_bd_cell -type hier myModule1
/myModule1
move_bd_cells /myModule1 [get_bd_cells /axi_fifo_1]
/myModule1
current_bd_instance /myModule1
/myModule1
create_bd_cell -type hier myModule2
/myModule1/myModule2
current_bd_instance
/
```

See Also

- copy_bd_objs
- current_bd_instance
- move_bd_cells
- update_module_reference

Send Feedback

create_bd_design

Create a new design and its top level hierarchy cell with the same name.

Syntax

```
create_bd_design [-dir <arg>] [-cell <arg>] [-quiet] [-verbose] <name>
```

Returns

The newly created design object, "" if failed.

Usage

Name	Description
[-dir]	Directory path for remote BD to be created and managed outside the project
[-cell]	hierarchical cell name which sub design to be copied into new design
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<name>	Name of design to create

Categories

IPIntegrator

Description

Create a new IP subsystem design module to add to the current project, and for use with the IP integrator of the Vivado Design Suite.

An empty IP subsystem module is created and added to the source files of the current project. The subsystem module and file are created with the specified `<name>` in the current project at:

```
<project_name>/<project_name>.srcs/sources_1/bd/<name>/<name>.bd
```

This command returns the file path and name of the IP subsystem design created if the command is successful. An error is returned if the command fails.

Arguments

`-dir <arg>` - (Optional) Specify the directory to write the block design file to. This lets you create and manage BD files outside of the current project directory structure, and facilitates reuse of block designs across multiple projects.

`-cell <arg>` - (Optional) Specify a hierarchical bd_cell to use as the source of a new block design. Use this option to create new block design from a portion of an existing design. The cell can be specified by instance name or returned as an object by the `get_bd_cells` command.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - The name of the IP subsystem design module to create.

Example

The following example creates a new empty IP subsystem module called `design_1`, adds the module to the current project, and creates a file called `design_1.bd` in the sources directory of the project:

```
create_bd_design design_1
```

See Also

- close_bd_design
- current_bd_design
- open_bd_design
- save_bd_design

create_bd_intf_net

Create a new intf_net.

Syntax

```
create_bd_intf_net [-quiet] [-verbose] <name>
```

Returns

The newly created intf_net object, "" if failed.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<name>	Name of intf_net to create

Categories

[IPIntegrator](#)

Description

Create a new IP integrator interface net for the subsystem design.

This command returns the newly created interface net object if successful, and returns noting if it failed.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - The names of interface net to create.

Example

The following example creates an interface net in the current subsystem design:

```
create_bd_intf_net diff_clock_rtl
```

See Also

- connect_bd_intf_net

Send Feedback

create_bd_intf_pin

Create a new intf_pin.

Syntax

```
create_bd_intf_pin [-vlnv <arg>] [-mode <arg>] [-quiet] [-verbose] [<name>]
```

Returns

The newly created intf_pin object, "" if failed.

Usage

Name	Description
[-vlnv]	Bus vlnv
[-mode]	Bus interface mode
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<name>]	Name of intf_pin to create

Categories

[IPIntegrator](#)

Description

Create a new interface pin on an IP integrator hierarchical module. An IP integrator interface is a grouping of signals that share a common function, and can include both individual signals and buses that share a related function. An AXI4-Lite master, for example, is an interface that includes a large number of individual signals plus multiple buses.

To create a single connection pin, or standard bus pin, use the `create_bd_pin` command.

Interface pins connect with other compatible interface pins, or interface ports. The interface pin is added as a port inside the hierarchical module, to connect outside of the module, and as a pin on the hierarchical module.

You must define the hierarchical module as the current instance in the IP integrator subsystem design, using the `current_bd_instance` command. The current instance is the target of the `create_bd_intf_pin` command.

This command returns the name of the newly created interface pin object if successful, and returns an error if it failed.

Send Feedback

Arguments

`-vlnv <arg>` - (Optional) The Vendor:Library:Name:Version (VLNV) attribute of the interface pin object to be added to the subsystem design. The VLNV attribute identifies the object in the IP integrator catalog.

Note: The `-vlnv` property for interface pins and ports refers to files in the Vivado Design Suite installation hierarchy. `-vlnv xilinx.com:interface:lmb_rtl:1.0` for example, is located in the Vivado Design Suite installation at `data/ip/interfaces/lmb_v1_0`.

`-mode <arg>` - (Optional) Defines the mode of the interface pin. Accepted values are Master, Slave, System, MirroredMaster, MirroredSlave, MirroredSystem, Monitor.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - (Required) The name of the interface pin to add to the current instance.

Example

The following example sets the hierarchical module, newMod1, as the current instance of the IP integrator subsystem design, and then creates a new interface pin on that module:

```
current_bd_instance [get_bd_cells /newMod1]
create_bd_intf_pin -mode Slave -vlnv xilinx.com:user:dma_rtl:1.0 data_in
```

See Also

- connect_bd_intf_net
- create_bd_port
- delete_bd_objs
- get_bd_intf_ports

create_bd_intf_port

Create a new interface port.

Syntax

```
create_bd_intf_port [-vlnv <arg>] [-mode <arg>] [-quiet] [-verbose]
    [<name>]
```

Returns

The newly created interface port object, "" if failed.

Usage

Name	Description
[-vlnv]	Bus vlnv
[-mode]	Bus interface mode
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<name>]	Name of port to create

Categories

IPIntegrator

Description

Create a new IP integrator subsystem design interface port. An IP integrator interface is a grouping of signals that share a common function, and can include both individual signals and buses that share a related function. An AXI4-Lite master, for example, is an interface that includes a large number of individual signals plus multiple buses.

To create a single connection port, or common bus port, use the `create_bd_port` command.

This command returns the name of the newly created interface port object if successful, and returns nothing if it failed.

Arguments

`-vlnv <arg>` - (Optional) The Vendor:Library:Name:Version (VLNV) attribute of the interface port object to be added to the subsystem design. The VLNV attribute identifies the object in the IP integrator catalog.

Send Feedback

Note: The `-vlnv` property for interface pins and ports refers to files in the Vivado Design Suite installation hierarchy that can be found in the `./data/ip/interfaces` directory. `-vlnv` `xilinx.com:interface:lmb_rtl:1.0`, for example, is located in the Vivado Design Suite installation at `data/ip/interfaces/lmb_v1_0`.

`-mode <arg>` - (Optional) Defines the mode of the interface pin. Accepted values are Master, Slave, System, MirroredMaster, MirroredSlave, MirroredSystem, Monitor.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - (Required) The name of the interface port to add to the subsystem design.

Example

The following example creates a new IP integrator interface port and adds it to the current subsystem design:

```
create_bd_intf_port -vlnv xilinx.com:interface:diff_clock_rtl:1.0 \
    -mode Slave diff_clock_rtl
```

See Also

- connect_bd_intf_net
- create_bd_intf_pin
- create_bd_port
- delete_bd_objs
- get_bd_intf_ports

Send Feedback

create_bd_intf_tlm_port

Create a new TLM interface port.

Syntax

```
create_bd_intf_tlm_port -vlnv <arg> -mode <arg> [-quiet] [-verbose] <name>
```

Returns

The newly created TLM interface port object, "" if failed.

Usage

Name	Description
-vlnv	TLM interface vlnv
-mode	TLM interface mode
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<name>	TLM interface name

Categories

[IPIntegrator](#)

create_bd_net

Create a new net.

Syntax

```
create_bd_net [-quiet] [-verbose] <name>
```

Returns

The newly created net object, "" if failed.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<name>	Name of net to create

Categories

[IPIntegrator](#)

Description

Create a new net in the current IP integrator subsystem design.

This command returns the newly created net object, or returns an error if failed.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - (Required) The names of nets to create. Net names can be specified from the top-level, as name only (net1), or can be specified within the design hierarchy by specifying the hierarchical net name (cell1/cellA/net1).

Example

The following example creates a new net:

```
create_bd_net net1
```

See Also

- connect_bd_intf_net
- connect_bd_net
- create_bd_cell
- create_bd_intf_net
- create_bd_intf_pin
- create_bd_intf_port
- create_bd_pin
- create_bd_port
- current_bd_design

create_bd_pin

Create a new pin.

Syntax

```
create_bd_pin [-from <arg>] [-to <arg>] -dir <arg> [-type <arg>] [-quiet]
    [-verbose] <name>
```

Returns

The newly created pin object, "" if failed.

Usage

Name	Description
[-from]	Begin index Default: Unspecified
[-to]	End index Default: Unspecified
-dir	Pin direction
[-type]	Pin type
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<name>	Name of pin to create

Categories

IPIntegrator

Description

Create a new pin to add to an IP integrator hierarchical module.

This command returns the name of the newly created pin object, or returns an error message if it failed.

Arguments

-from <arg> - (Optional) The starting index of a standard bus pin. This is unspecified for single bit pins.

-to <arg> - (Optional) The ending index of a standard bus pin. This is unspecified for single bit pins.

-dir [I | O | IO] - (Required) The direction of the pin. Valid values are I for input, O for output, and IO for bidirectional pins.

`-type <arg>` - (Optional) Defines the type of the pin as a clock pin (CLK), a reset pin (RST), a clock enable pin (CE), an interrupt pin (INTR), or as a data pin (DATA). If you do not define the pin type, it will be undefined (UNDEF).

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - (Required) The name of the subsystem pin to create. The name is referenced from the hierarchical module that the pin is being added to: `/modName/pinname`.

Examples

The following example creates a new input pin on the specified module in the current IP integrator subsystem design:

```
create_bd_pin -dir I -type rst /newMod1/rst
/newMod1/rst
```

See Also

- create_bd_cell
- create_bd_intf_pin
- create_bd_intf_port
- create_bd_port

create_bd_port

Create a new port for an IP subsystem design.

Syntax

```
create_bd_port [-from <arg>] [-to <arg>] -dir <arg> [-type <arg>]
    [-freq_hz <arg>] [-quiet] [-verbose] <name>
```

Returns

The newly created port object. Returns nothing if the command fails.

Usage

Name	Description
[-from]	Beginning index Default: Unspecified
[-to]	Ending index Default: Unspecified
-dir	Port direction. Valid values are I, O, or IO.
[-type]	Port type. Valid values are clk, ce, rst, intr, data.
[-freq_hz]	Frequency in Hertz for clock ports Default: Unspecified
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<name>	Name of port to create

Categories

[IPIntegrator](#)

Description

Create a new port to add to an IP integrator subsystem design. The port is a connection to signals external to the subsystem design.

This command returns the name of the newly created port object, or returns an error message if it failed.

Arguments

-from <arg> - (Optional) The starting index of a standard bus port. This is unspecified for single bit ports.

-to <arg> - (Optional) The ending index of a standard bus port. This is unspecified for single bit ports.

Send Feedback

`-dir [ I | O | IO ]` - (Required) The direction of the port. Valid values are I for input, O for output, and IO for bidirectional ports.

`-type <arg>` - (Optional) Defines the type of the port as a clock port (CLK), a reset port (RST), a clock enable port (CE), an interrupt port (INTR), or as a data port (DATA). If you do not define the port type, it will be undefined (UNDEF).

`-freq_hz <arg>` - (Optional) Specifies the frequency for clock ports as defined by the `-type` option. Unit is specified as Hz, so a 150 MHz clock would be 150000000. If not specified the clock is set to 100 MHz.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - (Required) The name of the subsystem port to create.

Examples

The following example creates a new bidirectional bus port in the current IP integrator subsystem design:

```
create_bd_port -from 0 -to 32 -dir IO -type data addr
/addr
```

The following example creates a new 125 MHz clock port:

```
create_bd_port -dir I -type clk -freq_hz 125000000 my_clock
```

See Also

- create_bd_cell
- create_bd_intf_pin
- create_bd_intf_port

create_bd_tlm_port

Create a new TLM port for an IP subsystem design.

Syntax

```
create_bd_tlm_port [-quiet] [-verbose] <name>
```

Returns

The newly created TLM port object. Returns nothing if the command fails.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<name>	Name of port to create

Categories

IPIntegrator

Send Feedback

create_cell

Create cells in the current design

Syntax

```
create_cell -reference <arg> [-black_box] [-quiet] [-verbose] <cells>...
```

Usage

Name	Description
-reference	Library cell or design which cells reference
[-black_box]	Create black box instance
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<cells>	Names of cells to create

Categories

Netlist

Description

Add cells to the netlist of the current Synthesized or Implemented design.

Note: You cannot add cells to library macros, or macro-primitives.

New cell instances can be added to the top-level of the design, or hierarchically within any module of the design. Instances can reference an existing cell from the library or design source files, or a black box instance can be added that reference cells that have not yet been created.

Netlist editing changes the in-memory view of the netlist in the current design. It does not change the files in the source fileset, or change the persistent design on the disk. Changes made to the netlist may be saved to a design checkpoint using the `write_checkpoint` command, or may be exported to a netlist file such as Verilog, VHDL, or EDIF, using the appropriate `write_*` command.

Note: Netlist editing is not allowed on the elaborated RTL design.

This command returns the name of the created cell instance or instances.

Arguments

`-reference <arg>` - (Required) The library cell or source file module referenced by the new cell instances.

Send Feedback

`-black_box` - (Optional) Define a black box instance of the specified reference cell. Use this argument when the reference cell does not exist yet, but you would like to create a black box instance of the cell for a top-down design approach.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<cells>` - (Required) Instance names of the cells to create. The instance name can be specified as a hierarchical name, from the top-level of the design. In this case, you must use the hierarchy separator character in the hierarchical instance name. You can determine the current hierarchy separator with the `get_hierarchy_separator` command.

Examples

The following example creates three new cell instances of the or1200_cpu module with the specified instance names:

```
create_cell -reference or1200_cpu myCell1 myCell2 myCell3
```

The following example sets the hierarchy separator character, then creates a black box instance for the referenced cell, specifying a hierarchical instance name:

```
set_hierarchy_separator |
create_cell -reference dmaBlock -black_box usbEngine0|myDMA
```

Note: The tool will return an error when `-black_box` is used, but the `-reference` cell already exists.

See Also

- connect_net
- create_net
- create_pin
- create_port
- remove_cell
- rename_cell
- set_hierarchy_separator
- write_checkpoint
- write_edif
- write_verilog

- [write_vhdl](#)

create_clock

Create a clock object

Syntax

```
create_clock -period <arg> [-name <arg>] [-waveform <args>] [-add] [-quiet]
    [-verbose] [<objects>]
```

Returns

New clock object.

Usage

Name	Description
`-period`	Clock period: Value > 0
`[-name]`	Clock name
`[-waveform]`	Clock edge specification
`[-add]`	Add to the existing clock in source_objects
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`[<objects>]`	List of clock source ports, pins or nets

Categories

SDC, XDC

Description

> **TIP:** *The XDC > Timing Constraints language templates and the Timing Constraints Wizard in the Vivado IDE offer timing diagrams and additional details around defining specific timing constraints. You can refer to these sources for additional information.*

Create a clock object with the specified period or waveform defined in nanoseconds (ns). This command defines primary clocks which are used by the timing engine as the delay propagation starting point of any clock edge. The defined clock can be added to the definition of an existing clock, or overwrite the existing clock.

A virtual clock can be created that has no source in the design. A virtual clock can be used as a time reference for setting input and output delays but does not physically exist in the design.

A clock can also be generated from an existing physical clock, and derive many of its properties from the master clock. Use the `create_generated_clock` command to derive a clock from an existing physical clock.

Send Feedback

> ⭐ **IMPORTANT!** *If you use* `create_clock` *to create a generated clock, instead of* `create_generated_clock`, *the created clock does not inherit any of the properties of its source clock. The insertion delay and jitter of the parent clock will not be propagated to the generated clock, causing incorrect timing calculations.*

The `create_clock` command returns the name of the clock object that is created.

Arguments

`-period <arg>` - (Required) Specifies the clock period of the clock object to be created. The value is specified as nanoseconds (ns) and must be greater than zero (>0).

`-name <arg>` - (Optional) The name of the clock object to be created. If the name is omitted, a system-generated name will be used based on the specified source `<objects>`. You can also use the `-name` option without source `<objects>` to create a virtual clock for the design that is not associated with a physical source on the design.

`-waveform` <arg1 arg2 ...> - (Optional) The rising and falling edge times of the waveform of the defined clock, in nanoseconds, over one full clock cycle. You can use multiple rising and falling edges to define the characteristics of the waveform, but there must be an even number of edges, representing both the rising and falling edges of the waveform. The first time specified (`arg1`) represents the time of the first rising transition, and the second time specified (`arg2`) is the falling edge. If the value for the falling edge is smaller than the value for the rising edge, it means that the falling edge occurs before the rising edge.

Note: If you do not specify the waveform, the default waveform is assumed to have a rising edge at time 0.0 and a falling edge at one half the specified period (`-period/2`).

`-add` - (Optional) Define multiple clocks on the same source for simultaneous analysis with different clock waveforms. Use `-name` to specify the new clock to add. If you do not specify this option, the `create_clock` command will overwrite any existing clock of the same name.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<objects>` - (Optional) The ports, pins, or nets which are the source of the specified clock. If you specify a clock on a source object that already has a clock, the new clock will overwrite the original clock unless you also specify the `-add` option. If no `<objects>` are specified to attach the clock object to, the clock will be created as a virtual clock in the design.

Note: The first driver pin of a specified net will be used as the source of the clock.

Examples

The following example creates a physical clock called bftClk and defines the clock period:

```
create_clock -name bftClk -period 5.000 [get_ports bftClk]
```

Note: If the `get_ports` command defining the objects is left off of this example, a virtual clock is created in the design rather than a physical clock.

The following example creates a clock named clk on the input port, bftClk, with a period of 10ns, the rising edge at 2.4ns and the falling edge at 7.4ns:

```
create_clock -name clk -period 10.000 -waveform {2.4 7.4} [get_ports bftClk]
```

The following example creates a virtual clock since no clock source is specified:

```
create_clock -name virtual_clock -period 5.000
```

The following example creates a clock with the falling edge at 2ns and the rising edge at 7ns:

```
create_clock -name clk -period 10.000 -waveform {7 2} [get_ports bftClk]
```

Note: Because the falling edge is earlier than the rising edge in the `-waveform` definition, although it is specified as `arg2`, it occurs first in the waveform.

See Also

- all_clocks
- create_generated_clock
- get_clocks
- report_clocks
- report_clock_interaction
- report_clock_networks
- report_clock_utilization
- set_clock_groups
- set_clock_latency
- set_clock_uncertainty
- set_input_delay
- set_output_delay
- set_propagated_clock

Send Feedback

create_cluster_configuration

Creates a cluster configuration

Syntax

```
create_cluster_configuration -submit_cmd <arg> -kill_cmd <arg>
    [-monitor_cmd <arg>] -type <arg> [-force] [-quiet] [-verbose] <name>
```

Usage

Name	Description
-submit_cmd	Submit command of cluster configuration
-kill_cmd	Kill command of cluster configuration
[-monitor_cmd]	Monitor command of cluster configuration
-type	Type of cluster configuration
[-force]	Force to overwrite a cluster configuration if it already exists.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<name>	Name of cluster configuration

Categories

cluster_configuration

Description

Creates cluster configuration to be used for launching runs (synthesis and implementation). Cluster configuration is defined using name, submit command, kill command and type.

Cluster configuration allows user to create configurations for different clusters and are persisted in Vivado tool settings. So user has to create cluster configuration just once and use whenever required.

Currently, Vivado supports following clusters management tools.

- Load Sharing Facility (LSF)

- Sun Grid Engine (SGE)

- Simple Linux Utility For Resource Management (SLURM)

Send Feedback

Arguments

`<name>` - (Required) Name of the cluster configuration. Cluster configuration is identified using its name, hence name of the cluster configuration should be unique. You can get a list of existing cluster configurations with the `get_cluster_configurations` command.

`<submit_cmd>` - (Required) Submit command for this cluster configuration. This is the command that will be used to submit the job on the cluster.

`<kill_cmd>` - (Required) Kill command for this cluster configuration. This is the command that will be used to kill the job on the cluster. The kill command must take job-id as input to kill the job.

`<type>` - (Required) Type of cluster configuration. This is the type of cluster for the cluster configuration being created.

- `LSF` - Cluster of type "Load Sharing Facility"

- `SGE` - Cluster of type "Sun Grid Engine"

- `SLURM` - Cluster of type "Simple Linux Utility For Resource Management"

- `CUSTOM` - Lets you create a cluster configuration for a cluster type which is not supported by Vivado. **This is an experimental feature and the cluster configuration of this type will not be validated thoroughly. It is provided to allow you to add cluster configuration for non-supported cluster types.**

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example creates a cluster configuration named *lsf_cluster_1* of type *LSF*. The submit command is `bsub` and the kill command is `bkill`.

```
create_cluster_configuration lsf_cluster_1 bsub bkill LSF
```

Send Feedback

The following example creates a cluster configuration named *lsf_medium* of type *LSF*. The submit command is `bsub` with resource requirement string which looks for 64bit linux os with either rhel or centos distro and version of the os is 7 and the jobs will be submitted to medium queue. The kill command is `bkill`.

```
create_cluster_configuration lsf_medium {bsub -R "select[os == lin && type
== X86_64 && (osdistro == rhel || osdistro == centos) && (osver == ws7)]" -
N -q medium} bkill LSF
```

The following example creates a cluster configuration named *sge_conf* of type *SGE*. The submit command is `qsub` and the kill command is `qdel`.

```
create_cluster_configuration sge_conf qsub qdel SGE
```

The following example creates a cluster configuration named *slurm_conf* of type *SLURM*. The submit command is `sbatch` and the kill command is `scancel`.

```
create_cluster_configuration slurm_conf sbatch scancel SLURM
```

See Also

- get_cluster_configurations
- remove_cluster_configurations
- validate_cluster_configurations

create_dashboard_gadget

Create the Gadget for Project summary dashboard

Syntax

```
create_dashboard_gadget -name <arg> -type <arg> [-quiet] [-verbose]
```

Usage

Name	Description
-name	Name of the gadget
-type	Type of the gadget
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Project

Description

> ⭐ **IMPORTANT!** *This command is primarily intended to be used in the Vivado IDE, with the Project Summary, and the **Add Gadget** command.*

Create a new "gadget" for a dashboard, such as the Project Summary dashboard that lets you view different aspects of the synthesized or implemented design run. Aspects of the design include timing information, resource utilization, DRC and methodology violations, and power analysis.

A dashboard gadget has a series of properties that define the information displayed, and the way it is displayed. These properties can be set using the `set_property` command. To determine the properties on a gadget, you can use the following command:

```
report_property -all [get_dashboard_gadget <gadget_name>]
```

Properties of a dashboard gadget can be set using the `set_property` command. Some of the properties of a gadget that can be set include:

- TYPE: Indicates the type of information presented by the gadget as defined by the `-type` option.

- ROW: Indicates the row placement in the dashboard.

- COL: Indicates the column placement of the gadget in the dashboard.

Send Feedback

- REPORTS: specifies the reports associated with the gadget.

- RUN.STEP and TYPE: specifies the synthesis or implementation run step that the gadget applies to.

- VIEW.TYPE and ORIENTATION: Specifies the presentation of information as a graph or table, and indicates the orientation of the data.

Arguments

`-name <arg>` - (Required) Specify the name of the gadget to create.

`-type <arg>` - (Required) Specify the type of dashboard gadget to create. The type is specified as one of an enumerated set which includes the following types: DRC, Methodology, Power, Timing, and Utilization.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Example

The following example creates a power type gadget, and defines the reports and run stages the gadget applies to:

```
create_dashboard_gadget -name gadget_power -type power
set_property reports {impl_1#impl_1_route_report_power_0}
[get_dashboard_gadgets [list {gadget_power}]]
set_property active_reports {impl_1#impl_1_route_report_power_0}
[get_dashboard_gadgets [list {gadget_power}]]
set_property run.step all_stages [get_dashboard_gadgets [list
{gadget_power}]]
```

See Also

- delete_dashboard_gadgets
- get_dashboard_gadgets
- move_dashboard_gadget

create_debug_core

Create a new Integrated Logic Analyzer debug core

Syntax

```
create_debug_core [-quiet] [-verbose] <name> <type>
```

Returns

New debug_core object.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<name>	Name of the new debug core instance
<type>	Type of the new debug core

Categories

Debug, XDC

Description

Adds a new Integrated Logic Analyzer (ILA) debug core to an open netlist design in the current project. The ILA debug core defines ports for connecting nets to for debugging the design in the logic analyzer feature of the Vivado Design Suite available through the open_hw command.

ILA debug cores can be added to the RTL source files of the design using debug cores from the Xilinx IP catalog, or added to the netlist design after synthesis using this command. Refer to the *Vivado Design Suite User Guide: Vivado Programming and Debugging (UG908)* for more information on using ILA debug cores.

Note: A debug core can only be added to an open netlist design using this command.

The ILA core is created with a CLK port and a PROBE port by default. The CLK port defines the clock domain for the ILA core, and allows you to probe signals that are common to that domain. The CLK port only supports one clock signal, and so you must create a separate debug core for each clock domain. The PROBE port provides a probe point for nets marked for debug with the MARK_DEBUG property. The PROBE port offers multiple channels to probe multiple nets from a single ILA core.

Send Feedback

You can add new ports to an existing ILA core with the `create_debug_port` command, and connect signals to the ports using the `connect_debug_port` command.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - (Required) The name of the ILA debug core to add to the project.

`<type>` - (Required) The type of debug core to insert. Only the ILA debug core is currently supported in the Vivado tool. The ILA debug core simply adds another load onto a connected net without otherwise altering it.

Note: When the ILA core is added to the project, the tool also adds a Debug Hub core (labtools_xsdbm_v1) as a container for one or more ILA cores. However, you cannot directly add a Debug Hub to the project.

Examples

The following example opens the synthesis run, creating the specified netlist design name, and then creates a new ILA debug core in that design:

```
open_run -name netlist_1 synth_1
create_debug_core myCore ila
```

The properties of the debug core can be customized by using the `set_property` command as in the following example:

```
set_property C_DATA_DEPTH 2048 [get_debug_cores myCore]
```

This example marks a sequence of nets for debugging using the MARK_DEBUG property, creates a new debug core, connects the CLK port to the appropriate clock domain, and assigns the debug nets to the PROBE ports on the core:

```
set_property MARK_DEBUG true [get_nets [list {control_reg[0]}
{control_reg[1]} \
    {control_reg[2]} {control_reg[3]} {control_reg[4]} {control_reg[5]} \
    {control_reg[6]} {control_reg[7]} {control_reg[8]} {control_reg[9]} \
    {control_reg[10]} {control_reg[11]} {control_reg[12]} {control_reg[13]} \
    {control_reg[14]} {control_reg[15]} {control_reg[16]} {control_reg[17]} \
    {control_reg[18]} {control_reg[19]} {control_reg[20]} {control_reg[21]} \
    {control_reg[22]} {control_reg[23]} {control_reg[24]} {control_reg[25]} \
    {control_reg[26]} {control_reg[27]} {control_reg[28]} {control_reg[29]} \
```

```
    {control_reg[30]} {control_reg[31]}]]
create_debug_core u_ila_0 ila
set_property port_width 1 [get_debug_ports u_ila_0/CLK]
connect_debug_port u_ila_0/CLK [get_nets [list wbClk ]]
set_property port_width 32 [get_debug_ports u_ila_0/PROBE0]
connect_debug_port u_ila_0/PROBE0 [get_nets [list {control_reg[0]}
    {control_reg[1]} {control_reg[2]} {control_reg[3]} {control_reg[4]} \
    {control_reg[5]} {control_reg[6]} {control_reg[7]} {control_reg[8]} \
    {control_reg[9]} {control_reg[10]} {control_reg[11]} {control_reg[12]} \
    {control_reg[13]} {control_reg[14]} {control_reg[15]} {control_reg[16]} \
    {control_reg[17]} {control_reg[18]} {control_reg[19]} {control_reg[20]} \
    {control_reg[21]} {control_reg[22]} {control_reg[23]} {control_reg[24]} \
    {control_reg[25]} {control_reg[26]} {control_reg[27]} {control_reg[28]} \
    {control_reg[29]} {control_reg[30]} {control_reg[31]} ]]
```

See Also

- connect_debug_port
- create_debug_port
- delete_debug_core
- get_debug_cores
- implement_debug_core
- open_run
- report_debug_core
- report_property
- set_property

Send Feedback

create_debug_port

Create a new debug port

Syntax

```
create_debug_port [-quiet] [-verbose] <name> <type>
```

Returns

New debug_port object.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<name>	Name of the debug core instance
<type>	Type of the new debug port

Categories

[Debug](), [XDC]()

Description

Defines a new port to be added to an existing Vivado ILA debug core that was added to the design using the `create_debug_core` command. The port provides connection points on an ILA core to attach nets from the design for debugging.

When a new debug core is created using the `create_debug_core` command, it includes a `clk` and `probe` port by default. However, you can add trigger input/output port types as well. Refer to the *Vivado Design Suite User Guide: Programming and Debugging (UG908)* for more information on port types and purpose.

A port can have one or more connection points to support one or more nets to debug. As a default new ports are defined as having a width of 1, allowing only one net to be attached. You can change the port width of `probe` ports to support multiple signals using the `set_property port_width` command (see Examples).

Note: clk, trig_in, trig_in_ack, trig_out, and trig_out_ack ports can only have a width of 1.

You can connect signals to ports using the `connect_debug_port` command, modify existing probe connections using `modify_debug_ports`, and disconnect signals with the `disconnect_debug_port` command.

Send Feedback

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - (Required) The name of the ILA debug core to add the new port to. The debug core must already exist in the project having been created with `create_debug_core`.

`<type>` - (Required) The type of debug port to insert. The supported port types are:

- `clk` - Defines the clock port for connecting the ILA debug core to a clock domain. Each debug core can only have one `clk` port, and each `clk` port can only connect to one clock domain. Therefore you must use multiple ILA cores to probe signals from different clock domains.

  > ⭐ **IMPORTANT!** *Ensure that the connected clocks are free-running clocks. Failure to do so could result in an inability to communicate with the debug core when the design is loaded onto the device.*

- `probe` - Provides probe points to connect to signals that are marked for debugging with the MARK_DEBUG property. The ILA debug core can contain multiple `probe` ports, which are automatically numbered by the Vivado tool when the port is added to the core. Each `probe` port can contain one or more channels, or connection points, as defined by the PORT_WIDTH property.

- `trig_in`/`trig_in_ack`, and `trig_out`/`trig_out_ack` - The ILA probe trigger comparators used to detect specific comparison conditions on the `probe` inputs to the ILA core. `trig_in` and `trig_in_ack`, and `trig_out` and `trig_out_ack` should be added to the debug core as port pairs when used. Refer to the *Vivado Design Suite User Guide: Programming and Debugging (UG908)* for more information.

Examples

The following example creates a new debug core, and then adds an additional `probe` port to the core, then sets the width of that new port to 8, and connects signals to the `probe` port:

```
create_debug_core myCore ila
create_debug_port myCore probe
set_property PORT_WIDTH 8 myCore/probe1
connect_debug_port -channel_start_index 1 myCore/probe1 \
{m1_cyc_i m1_ack_o m1_err_o m1_rty_o}
```

Note: Recall that the ILA core is created with a clk and probe port by default, so the new probe port is automatically numbered as probe1.

Send Feedback

See Also

- connect_debug_port
- create_debug_core
- disconnect_debug_port
- set_property

create_drc_check

Create a user defined DRC rule

Syntax

```
create_drc_check [-hiername <arg>] -name <arg> [-desc <arg>] [-msg <arg>]
    -rule_body <arg> [-severity <arg>] [-quiet] [-verbose]
```

Usage

Name	Description
[-hiername]	Specify the hiername for this rule. When the DRC UI panel is created, this is used to place the new rule in the menu hierarchy. Use a dot (.) to separate layers in the menu hierarchy. It is optional and will default to User Defined. Default: User Defined
-name	Specify the name for this rule. This must be of the form PREFIX-id where XXXX is a 4-6 letter abbreviation and id is an integer identifying a particular rule. Similar checks should have the same abbreviation and each a unique id.
[-desc]	Specify the short description for this rule. It is optional and will default to <User rule - default description>. Default: User rule - default description
[-msg]	Specify the full description for this rule. Including the substitutions. Values are: %MSG_STRING %NETLIST_ELEMENT %SITE_GROUP %CLOCK_REGION %BANK %BEL_GROUP.
-rule_body	The string representing the body of the rule. This can be a tcl proc name or any string of tcl code to be evaluated.
[-severity]	Specify severity level for a DRC rule. Default: Warning. Values: Error, Critical Warning, Warning, Advisory.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[DRC](), [Object]()

Description

Create a new user-defined DRC rule check, drc_check, for use by the tool when running `report_drc`.

This command allows you to define a unique name or abbreviation for the user-defined rule check, optionally group the rule into a special hierarchy and provide a description of the rule, define a general placeholder message for the check when violations are encountered, and refer to the Tcl code associated with the design rule check to be run during the `report_drc` command.

Send Feedback

The general placeholder message defined in this command is populated with specific information related to the design objects and violations found by the Tcl checker procedure, and by the `create_drc_violation` command.

The process in brief is:

- Write a Tcl checker procedure to define the method applied when checking the user-defined rule, and the objects to check against the rule. The Tcl checker procedure is defined in a separate Tcl script that must be loaded by the `source` command prior to running `report_drc`.

- Use `create_drc_violation` in the Tcl checker to identify and flag violations found when checking the rule against a design.

- Define a user-defined DRC rule check using the `create_drc_check` command that calls the Tcl checker proc from the `-rule_body`.

- Create a rule deck using the `create_drc_ruledeck` command, and add the user-defined rule check to the rule deck using the `add_drc_checks` command.

- Run `report_drc`, and specify either the rule deck, or the user-defined rule check to check for violations.

If a drc_check of the specified name is already defined in the tool, an error is returned. In this case, to overwrite or redefine and existing drc_check, you must first delete the check using the `delete_drc_check` command.

The DRC rule check object features the `is_enabled` property that can be set to TRUE or FALSE using the `set_property` command. When a new rule check is created, the `is_enabled` property is set to TRUE as a default. Set the `is_enabled` property to FALSE to disable the rule check from being used when `report_drc` is run. This lets you create new DRC checks, add them to rule decks using `add_drc_checks`, and then enable them or disable them as needed without having to remove them from the rule deck.

Each user defined DRC rule check has the 'USER_DEFINED' property, which lets you quickly identify and select user-defined rule checks.

Arguments

`-hiername <arg>` - (Optional) Defines a rule grouping for the new rule. The default is "User Defined". This is used as the first level of hierarchy in the GUI when listing DRC rules. All newly created DRC checks are also added to the "all" hierarchy used by default by the `report_drc` command.

`-name <arg>` - (Required) The unique name for the design rule. This should match the name used by the `create_drc_violation` commands in the Tcl checker procedure specified in `-rule_body`. The name will appear in the DRC report with any associated violations. The name should consist of a short 4 to 6 letter abbreviation for the rule group, and an ID to differentiate it from other checks in the same group, for instance ABCD-1 or ABCD-23.

`-desc <arg>` - (Optional) A brief description of the rule. The default is "User Rule". This is displayed when listing DRC rules in the GUI. The description is also used in the DRC report and summary.

`-msg< arg>` - (Optional) This is the message displayed when a violation of the rule is found. The message can include placeholders for dynamic substitution with design elements found in violation of the rule. The design data is substituted into the message at the time `report_drc` is run. Each substitution key has a long form, and a short form as shown below. Valid substitutions keys are:

- %MSG_STRING (%STR) - This is the message string defined by the `-msg` option in the `create_drc_violation` command for the specific violation.

  Note: %STR is the default message for the `create_drc_check` command if the `-msg` option is not specified. In this case, any message defined by `create_drc_violation` in the `-rule_body` is simply passed through to the DRC report.

- %NETLIST_ELEMENT (%ELG) - Netlist elements including cells, pins, ports, and nets.

- %SITE_GROUP (%SIG) - Device site.

- %CLOCK_REGION (%CRG) - Clock region.

- %BANK (%PBG) - Package IO bank.

`-rule_body <arg>` - (Required) This is the name of the Tcl procedure which defines the rule checking functionality. The Tcl procedure can be embedded here, into the `-rule_body` option, or can be separately defined in a Tcl script that must be loaded with the `source` command when the tool is launched, or prior to running the `report_drc` command.

The Tcl checker procedure can create DRC violation objects, using the `create_drc_violation` command, containing the design elements that are associated with a design rule violation. The tool populates the substitution keys in the message defined by `-msg` with the design elements from the violation object.

`-severity <arg>` - (Optional) Specifies the severity of the rule being created. The default value is Warning. The possible values are:

- ERROR

- "CRITICAL WARNING"

- WARNING

- ADVISORY

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example defines a new design rule check named RAMW-1, with the hierarchy name and description defined, using the default severity of Warning, and calling the dataWidthCheck procedure when the check is run:

```
create_drc_check -name {RAMW-1} -hiername {RAMB} \
  -desc {Data Width Check} -rule_body dataWidthCheck -severity Advisory
```

The following Tcl script defines the `dataWidthCheck` procedure which is called by the `-rule_body` argument of the RAMW-1 check. This Tcl script file must be loaded into the tool using the `source` command, prior to running the `report_drc` command.

```
# This is a simplistic check -- report BRAM cells with WRITE_WIDTH_B
# wider than 36.
proc dataWidthCheck {} {
  # list to hold violations
  set vios {}
  # iterate through the objects to be checked
  foreach bram [get_cells -hier -filter {PRIMITIVE_SUBGROUP == bram}] {
    set bwidth [get_property WRITE_WIDTH_B $bram]
    if { $bwidth > 36} {
      # define the message to report when violations are found
      set msg "On cell %ELG, WRITE_WIDTH_B is $bwidth"
      set vio [ create_drc_violation -name {RAMW-1} -msg $msg $bram ]
      lappend vios $vio
    }
  }
  if {[llength $vios] > 0} {
    return -code error $vios
  } else {
    return {}
  }
}
create_drc_check -name {RAMW-1} -hiername {RAMB Checks} \
    -desc {Data Width Check} -rule_body dataWidthCheck \
    -severity Advisory
```

Note: The script file can contain both the Tcl checker procedure, and the `create_drc_check` command that defines it for use by `report_drc` command. In this case, when the Tcl script file is sourced, both the `dataWidthCheck` proc and the RAMW-1 design rule check are loaded into the tool.

See Also

- add_drc_checks
- create_drc_ruledeck
- create_drc_violation
- delete_drc_check

- get_drc_checks
- get_drc_violations
- report_drc

create_drc_ruledeck

Create one or more user defined DRC rule deck objects

Syntax

```
create_drc_ruledeck [-quiet] [-verbose] <ruledecks>...
```

Returns

Drc_ruledeck

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<ruledecks>	Names of DRC rule decks to create

Categories

DRC, Object

Description

Create one or more user-defined rule decks for use when running `report_drc`.

A drc_ruledeck object is a collection of design rule checks, grouped for convenience, to be run at different stages of the FPGA design flow, such as during I/O planning or placement. The tool comes with a set of factory predefined rule decks. Use the `get_drc_ruledecks` command to return a list of the currently defined rule decks.

The rule decks created by this command are empty, without any checks. You must add design rule checks to the rule deck using the `add_drc_checks` command. Checks can be removed from a rule deck using the `remove_drc_checks` command. To see a list of design rule checks that are available to include in the ruledeck, use the `get_drc_checks` command.

This command returns the list of drc_ruledecks created.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Send Feedback

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<ruledecks>` - (Required) Specify the name of one or more user-defined DRC rule decks to create.

Examples

The following example creates two new drc_ruledeck objects:

```
create_drc_ruledeck my_rules project_rules
```

See Also

- add_drc_checks
- delete_drc_ruledeck
- get_drc_checks
- get_drc_ruledecks
- remove_drc_checks
- report_drc

create_drc_violation

Create a DRC violation

Syntax

```
create_drc_violation -name <arg> [-severity <arg>] [-msg <arg>] [-quiet]
    [-verbose] [<objects>...]
```

Usage

Name	Description
`-name`	Specify the name for this rule. This is the typically a 4-6 letter specification for your rule.
`[-severity]`	Specify severity level for a DRC rule. Default: WARNING. Values: FATAL, ERROR, CRITICAL WARNING, WARNING, ADVISORY.
`[-msg]`	Specify your message string for this DRC rule.
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`[<objects>]`	Cells, ports, pins, nets, clock regions, sites, package banks to query.

Categories

[DRC](#), [Report](#)

Description

Create a DRC violation object and manage the list of design objects associated with the violation for reporting by the `report_drc` command.

The `create_drc_violation` command is specified as part of the Tcl checker procedure that defines and implements the checking feature of a user-defined design rule check created by the `create_drc_check` command. A violation object is created by the Tcl checker each time a violation of the design rule is encountered.

The process in brief is:

- Write a Tcl checker procedure to define the method applied when checking the user-defined rule, and the objects to check against the rule. The Tcl checker procedure is defined in a separate Tcl script that must be loaded by the `source` command prior to running `report_drc`.

- Use `create_drc_violation` in the Tcl checker to identify and flag violations found when checking the rule against a design.

- Define a user-defined DRC rule check using the `create_drc_check` command that calls the Tcl checker proc from the `-rule_body`.

- Create a rule deck using the `create_drc_ruledeck` command, and add the user-defined rule check to the rule deck using the `add_drc_checks` command.

- Run `report_drc`, and specify either the rule deck, or the user-defined rule check to check for violations.

Violations are reported by the `report_drc` command, and violation objects can be returned by the `get_drc_violations` command. The design objects associated with a DRC violation object can be obtained using the `-of_objects` option of the appropriate `get_*` command, such as `get_cells`, `get_nets`, or `get_ports` for instance:

```
get_ports -of_objects [get_drc_violations -name drc_1 NSTD*]
```

Arguments

`-name <arg>` - (Required) The name of the design rule check associated with the violation. This should be the same name used by the `create_drc_check` command which calls the associated Tcl checker procedure from its `-rule_body` argument. Messages from the `create_drc_violation` command are passed up to the drc_check with the same `-name`.

`-severity <arg>` - (Optional) The severity of the created violation. This allows individual DRC violations to override the default severity of a specific rule check. The default severity for user-defined DRCs is determined by the definition of `-severity` in the `create_drc_check` command. The supported values are:

- ERROR

- "CRITICAL WARNING"

- WARNING

- ADVISORY

Note: The `SEVERITY` is stored as a property on the DRC rule associated with the DRC violation object.

`-msg <arg>` - (Optional) This is a violation specific message that is substituted for the general string variable (%STR) specified in the optional placeholder message defined in the `create_drc_check` command.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Send Feedback

Note: Message limits can be defined with the `set_msg_config` command.

`<objects>` - (Optional) Cell, port, pin, net, clock region, site, and package I/O bank objects associated with violations found by the Tcl checker procedure that are substituted into the placeholder message of the drc_object with the same `-name`. Design objects map to substitution keys in the message as follows:

- %ELG - netlist elements such as cells, ports, pins, and nets.

- %CRG - clock regions.

- %SIG - device sites.

- %PBG - package I/O banks.

Note: Both the order and the type of `<objects>` passed from the `create_drc_violation` command must match the `-msg` specification from the `create_drc_check` command, or the expected substitution will not occur.

Examples

The following Tcl script defines the `dataWidthCheck` procedure which is called by the `-rule_body` argument of the RAMW-1 check. This Tcl script file must be loaded into the tool using the `source` command, prior to running the `report_drc` command.

Some features of the Tcl checker proc to notice are:

- A list variable is created to store violations (`$vios`)

- A violation object is created, and added to the list variable, each time a violation is found.

- The placeholder key `%ELG` in the `$msg` string is dynamically substituted with the specific `$bram` cell associated with the violation.

- The `dataWidthCheck` proc returns an error code when any violations are found (`$vios >0`) to inform the `report_drc` command of the results of the check.

- The list of violations is passed along with the return code, and the violations are reported by `report_drc`.

```
# This is a simplistic check -- report BRAM cells with WRITE_WIDTH_B
# wider than 36.
proc dataWidthCheck {} {
  # list to hold violations
  set vios {}
  # iterate through the objects to be checked
  foreach bram [get_cells -hier -filter {PRIMITIVE_SUBGROUP == bram}] {
    set bwidth [get_property WRITE_WIDTH_B $bram]
    if { $bwidth > 36} {
      # define the message to report when violations are found
      set msg "On cell %ELG, WRITE_WIDTH_B is $bwidth"
      set vio [ create_drc_violation -name {RAMW-1} -msg $msg $bram ]
      lappend vios $vio
    }
```

```
  }
  if {[llength $vios] > 0} {
    return -code error $vios
  } else {
    return {}
  }
}
create_drc_check -name {RAMW-1} -hiername {RAMB Checks} \
    -desc {Data Width Check} -rule_body dataWidthCheck \
    -severity Advisory
```

Note: The script file can contain both the Tcl checker procedure, and the `create_drc_check` command that defines it for use by `report_drc` command. In this case, when the Tcl script file is sourced, both the `dataWidthCheck` proc and the RAMW-1 design rule check are loaded into the tool.

See Also

- add_drc_checks
- create_drc_ruledeck
- create_drc_check
- get_cells
- get_drc_checks
- get_drc_violations
- get_nets
- get_pins
- get_ports
- get_sites
- report_drc
- set_property

Send Feedback

create_fileset

Create a new fileset

Syntax

```
create_fileset [-constrset] [-simset] [-blockset] [-clone_properties <arg>]
    -define_from <arg> [-quiet] [-verbose] <name>
```

Returns

New fileset object.

Usage

Name	Description
`[-constrset]`	Create fileset as constraints fileset (default)
`[-simset]`	Create fileset as simulation source fileset
`[-blockset]`	Create fileset as block source fileset
`[-clone_properties]`	Fileset to initialize properties from
`-define_from`	Name of the module in the source fileset to be the top of the blockset
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<name>`	Name of the fileset to be create

Categories

Project, Simulation

Description

Defines a new fileset within a design project. Files can be added to a newly created fileset using the `add_files` command.

A fileset is a list of files with a specific function within the project. One or more constraint files is a constraint set (`-constrset`); one or more simulation test benches is a simulation set (`-simset`). Only one fileset option can be specified when using the `create_fileset` command. As a default, the tool will create a constraint fileset if the type is not specified.

You can also use the `create_fileset -blockset` command to configure an IP core, or hierarchical module of the design, as an out-of-context (OOC) block. The block fileset, or blockset, creates a hierarchical file collection for the IP or module specified with the `-define_from` option. The files related to the specified hierarchical module will be moved from their current fileset to the new blockset. When the blockset is created, the Vivado Design Suite

Send Feedback

also defines out-of-context synthesis and implementation runs for the block. The output products for the OOC module are stored in the blockset, including the synthesized design checkpoint (DCP) and any required structural simulation netlists. Stuctural simulation netlists are needed when a behavioral model for the block is not available, or is not available in the language supported by the target simulator. You can define an out-of-context constraint file for the IP or moduleif needed, and add the at to the block fileset as well.

> **TIP:** *Refer to the Vivado Design Suite User Guide: Designing with IP (UG896) or the Vivado Design Suite User Guide: Hierarchical Design (UG905) for more information on out-of-context design.*

The `create_fileset` command returns the name of the newly created fileset, or will return an error message if it fails.

Arguments

`-constrset` - (Optional) Creates a constraint set to hold one or more constraint files. This is the default fileset created if neither the `-constrset`, `-simset`, or `-blockset` argument is specified.

`-simset` - (Optional) Create a simulation fileset to hold one or more simulation source files. You can only specify one type of fileset argument, either `-constrset` or `-simset`. You will get an error if both are specified.

`-blockset` - (Optional) Create a block fileset to configure an IP core or hierarchical module for out-of-context design.

> **IMPORTANT!** *The `-blockset` option requires the `-define_from` option to specify the IP or module to use as the top-level of the blockset.*

`-clone_properties <arg>` - (Optional) Clone the properties of a specified fileset to add to the newly created fileset. This is useful for ensuring that new filesets are created with needed properties such as USED_IN.

`-define_from <arg>` - (Optional) Specify the top-module of an IP core to define the block fileset.

> **IMPORTANT!** *This option is required when the `-blockset` option is used.*

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - (Required) The name of the fileset to be created.

Examples

The following example creates a new constraint file set named constraints2:

```
create_fileset -constrset -quiet constraints2
```

Note: With `-quiet` specified, the tool will not return anything if it encounters an error in trying to create the specified fileset.

The following example creates an out-of-context (OOC) blockset for the hierarchical module specified by the `-define_from` option:

```
create_fileset -blockset -define_from dac_spi dac_spi
```

The following example creates a new simulation fileset named `sim_1`:

```
create_fileset -simset sim_1
```

See Also

- add_files
- current_fileset
- synth_ip

create_generated_clock

Create a generated clock object

Syntax

```
create_generated_clock [-name <arg>] [-source <args>] [-edges <args>]
    [-divide_by <arg>] [-multiply_by <arg>] [-combinational]
    [-duty_cycle <arg>] [-invert] [-edge_shift <args>] [-add]
    [-master_clock <args>] [-quiet] [-verbose] <objects>
```

Returns

New clock object.

Usage

Name	Description
[-name]	Generated clock name
[-source]	Master clock source object pin/port
[-edges]	Edge Specification
[-divide_by]	Period division factor: Value >= 1 Default: 1
[-multiply_by]	Period multiplication factor: Value >= 1 Default: 1
[-combinational]	Create a divide_by 1 clock through combinational logic
[-duty_cycle]	Duty cycle of clock period: Range: 0.0 to 100.0 Default: 50.0
[-invert]	Invert the signal
[-edge_shift]	Edge shift specification
[-add]	Add to the existing clock in source_objects
[-master_clock]	Use this clock if multiple clocks present at master pin
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<objects>	List of clock source ports, pins, or nets

Categories

SDC, XDC

Description

💡 **TIP:** *The XDC > Timing Constraints language templates and the Timing Constraints Wizard in the Vivado IDE offer timing diagrams and additional details around defining specific timing constraints. You can refer to these sources for additional information.*

Generate a new clock object from an existing physical clock object in the design.

Send Feedback

Clocks can be added to a design in one of three ways:

- Primary physical or virtual clocks defined with the `create_clock` command.

- Derived clocks defined with the `create_generated_clock` command generated from a primary physical clock.

- Derived clocks automatically generated by the Vivado Design Suite when a clock propagates through an MMCM/PLL/BUFR.

You can also use the `create_generated_clock` command to change the name of clocks that the Vivado tool has auto-derived from an MMCM/PLL/BUFR. In this case, a new clock is not created, but an existing clock defined on the specified source object is renamed to the provided name. This requires `-name` and `<object>` to be specified, and supports the use of `-source` and/or `-master_clock` to further identify the clock to rename when multiple clocks exist on the source object. Refer to the *Vivado Design Suite User Guide: Using Constraints (UG903)* for more information on renaming auto-derived clocks.

> ⭐ **IMPORTANT!** *You cannot rename a clock that is already in use by other constraints at the time of renaming. You must rename the clock prior to any other appearance or use of the clock in an XDC file.*

This command returns the name of the clock object that is created, or returns an error if it fails.

Arguments

`-name <arg>` - (Optional) The name of the generated clock to create on the specified object, or the name to assign to an existing clock on the specified object. If no name is specified, the generated clock will be given the name of the `<object>` it is assigned to. If assigned to multiple `<objects>`, the name will be the first object in the list.

`-source <arg>` - (Optional) The pin or port of the master clock from which to derive the generated clock. The master clock must be a previously defined physical clock, not a virtual clock; but can be a primary clock or another generated clock. If the source pin or port currently has multiple clocks defined, the `-master_clock` option must be used to identify which clock on the source is to be used to define the generated clock.

`-edges <arg>` - (Optional) Specifies the edges of the master clock to use in defining transitions on the generated clock. Specify transitions on the generated clock in a sequence of 1, 2, 3, by referencing the appropriate edge count from the master clock in numerical order, counting from the first edge. The sequence of transitions on the generated clock defines the period and duty cycle of the clock: position 1 is the first rising edge of the generated clock, position 2 is the first falling edge of the generated clock and so defines the duty cycle, position 3 is the second rising edge of the generated clock and so defines the clock period. Enclose multiple edge numbers in braces {}. See the example below for specifying edge numbers.

`-divide_by <arg>` - (Optional) Divide the frequency of the master clock by the specified value to establish the frequency of the generated clock object. The value specified must be >= 1, and must be specified as an integer.

Send Feedback

`-multiply_by <arg>` - (Optional) Multiply the frequency of the master clock by the specified value to establish the frequency of the generated clock object. The value specified must be >= 1, and must be specified as an integer. The default is 1.

`-combinational` - (Optional) Calculate the generated clock latency by tracing delays through only the combinational paths between the source pin of the generated clock and the source pin of the master clock. By default the tool computes the latency for the generated clock by tracing both sequential and combinational paths.

`-duty_cycle <arg>` - (Optional) The duty cycle of the generated clock defined as a percentage of the new clock period when used with the `-multiply_by` argument. The value is specified as a percentage from 0.0 to 100. The default value is 50.0.

`-invert` - (Optional) Create a generated clock with the phase inverted from the master clock.

`-edge_shift <arg>` - (Optional) Shift the edges of the generated clock by the specified values relative to the master clock. See the example below for specifying edge shift.

`-add` - (Optional) Define multiple clocks on the same source for simultaneous analysis with different clock waveforms. Use `-name` to specify the new clock to add. If you do not specify this option, the `create_clock` command will automatically assign a name and will overwrite any existing clock of the same name.

Note: `-master_clock` and `-name` options must be specified with `-add`.

`-master_clock <arg>` - (Optional) If there are multiple clocks found on the source pin or port, the specified clock object is the one to use as the master for the generated clock object.

Note: `-add` and `-name` options must be specified with `-master_clock`.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<objects>` - (Required) The pin or port objects to which the generated clock should be assigned. If the specified objects already have a clock defined, use the `-add` option to add the new generated clock and not overwrite any existing clocks on the object.

Examples

The following example defines a generated clock that is divided from the master clock found on the specified CLK pin. Since `-name` is not specified, the generated clock is assigned the same name as the pin it is assigned to:

```
create_generated_clock -divide_by 2 -source \
    [get_pins clkgen/cpuClk] [get_nets fftEngine/CLK]
```

The following example defines a generated clock named CLK1 from the specified source clock, specifying the edges of the master clock to use as transition points for the generated clock, with edges shifted by the specified amount. In this example, the `-edges` option indicates that the second edge of the source clock is the first rising edge of the generated clock, the third edge of the source clock is the first falling edge of the generated clock, and the eighth edge of the source clock is the second rising edge of the generated clock. These values determine the period of the generated clock as the time from edge 2 to edge 8 of the source clock, and the duty cycle as the percentage of the period between edge 2 and edge 3 of the source clock. In addition, each edge of the generated clock is shifted by the specified amount:

```
create_generated_clock -name CLK1 -source CMB/CLKIN -edges {2 3 8} \
-edge_shift {0 -1.0 -2.0} CMB/CLKOUT
```

Note: The waveform pattern of the generated clock is repeated based on the transitions defined by the `-edges` argument.

This example creates two generated clocks from the output of a MUX, using `-master_clock` to identify which clock to use, using `-add` to assign the generated clocks to the Q pin of a flip flop, and using `-name` to define a name for the generated clock, since the object it is assigned to has multiple clocks assigned:

```
create_generated_clock -source [get_pins muxOut] -master_clock M_CLKA \
-divide_by 2 -add -name gen_CLKA [get_pins flop_Q]
create_generated_clock -source [get_pins muxOut] -master_clock M_CLKB \
-divide_by 2 -add -name gen_CLKB [get_pins flop_Q]
```

The following example renames the automatically named clock that is derived by the Vivado Design Suite on the MMCM clock output:

```
create_generated_clock -name CLK_DIV2 [get_pins mmcm/CLKOUT1]
```

See Also

- check_timing
- create_clock
- get_generated_clocks
- get_pins
- report_clocks
- set_clock_latency

- set_clock_uncertainty
- set_propagated_clock

Send Feedback

create_gui_custom_command

Create a custom command in the GUI

Syntax

```
create_gui_custom_command -name <arg> [-menu_name <arg>]
    [-description <arg>] [-show_on_toolbar] [-run_proc <arg>]
    [-toolbar_icon <arg>] [-command <arg>] [-tcl_file <arg>] [-quiet]
    [-verbose]
```

Usage

Name	Description
-name	Unique name of the command to create.
[-menu_name]	Menu name for the custom command.
[-description]	Display this text for the description of the menu item and optionally the toolbar button
[-show_on_toolbar]	Add this command to the toolbar
[-run_proc]	Needed when '-command' and 'tcl_file' options are both specified. If true, gui button will run command mentioned in '-command' option otherwise source script mentioned in '-tcl_file' option
[-toolbar_icon]	The full path to the PNG or JPEG file to display on the toolbar button
[-command]	The command to execute
[-tcl_file]	The full path to the Tcl file to source
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

GUIControl

Description

This command creates a GUI menu item for a user custom command. The menu command appears in the **Tools → Custom Commands** menu in the Vivado IDE.

The custom GUI command can also be shown/hidden on the toolbar menu. When the GUI button is clicked the custom command is executed. A Tcl file can also be sourced when the GUI button is clicked.

You can use `get_gui_custom_commands` to determine the list of user-defined custom commands.

Send Feedback

Arguments

`-name` - (Required) Unique name of the command to create.

`-menu_name` - (Optional) Menu name for the custom command.

`-description` - (Optional) Display this text for the description of the menu item and optionally the toolbar button

`-show_on_toolbar` - (Optional) Add this command to the toolbar

`-run_proc` - (Optional) Needed when both `-command` and `-tcl_file` options are specified. If TRUE, the menu command will run the specified `-command`, otherwise source the Tcl script specified by `-tcl_file`.

`-toolbar_icon` - (Optional) The full path to the PNG or JPEG file to display on the toolbar button

`-command` - (Optional) The command to be executed.

`-tcl_file` - (Optional) The full path to the Tcl file to source.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example creates a GUI custom command with name 'print_version', and adds it to the toolbar.

```
create_gui_custom_command -name print_version -command "version" \
-description "Gets tool version" -show_on_toolbar
```

See Also

- create_gui_custom_command_arg
- get_gui_custom_command_args
- get_gui_custom_commands
- remove_gui_custom_command_args
- remove_gui_custom_commands

Send Feedback

create_gui_custom_command_arg

Create a custom command argument for a custom command in the GUI

Syntax

```
create_gui_custom_command_arg -command_name <arg> -arg_name <arg>
    [-default <arg>] [-comment <arg>] [-optional] [-quiet] [-verbose]
```

Usage

Name	Description
-command_name	Unique name of the custom command for which an argument is being created.
-arg_name	Unique name of the custom command argument to create.
[-default]	Default value of the custom command argument.
[-comment]	Comment for the custom command argument.
[-optional]	Make the custom command argument optional.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[GUIControl](#)

Description

This commands defines an argument for a specific custom GUI command.

You can use `get_gui_custom_commands` to determine the list of defined custom commands. You can use `get_gui_custom_command_args` to determine the list of defined GUI custom command arguments of a particular GUI custom command.

Arguments

`-command_name` - (Required) The name of the custom GUI command to create an argument for.

`-arg_name` - (Required) The unique name of the argument to create.

`-default` - (Optional) Specifies a default value for the argument.

`-comment` - (Optional) Comment for the custom command argument.

`-optional` - (Optional) Make the custom command argument optional.

Send Feedback

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

This example creates an argument named 'quiet' for the custom GUI command `print_version`. The argument is defined as optional and its default value is `-quiet`.

```
create_gui_custom_command_arg -command_name print_version -arg_name quiet \
-default "-quiet" -comment "Ignore commands errors" -optional
```

See Also

- create_gui_custom_command
- get_gui_custom_command_args
- get_gui_custom_commands
- remove_gui_custom_command_args
- remove_gui_custom_commands

Send Feedback

create_hw_axi_txn

Create hardware AXI transaction object

Syntax

```
create_hw_axi_txn [-address <arg>] [-data <arg>] [-size <arg>] -type <arg>
    [-len <arg>] [-burst <arg>] [-cache <arg>] [-id <arg>] [-force]
    [-quiet] [-verbose] <name> <hw_axi>
```

Returns

New hardware AXI transaction object.

Usage

Name	Description
[-address]	AXI read or write address. Default: Address zero
[-data]	Transaction data. Default: All zeroes
[-size]	Deprecated. Data word size in bits. This is now automatically set based on the IP core properties.
-type	READ or WRITE transaction.
[-len]	Length of the transaction in data words. Default: 1
[-burst]	Burst type: INCR,FIXED or WRAP. Default: INCR
[-cache]	AXI cache type. Default: 3
[-id]	Address ID. Default: 0
[-force]	Overwrite an existing transaction with the specified name if it exists, otherwise create a new transaction. Default: 0
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<name>	Name of new object.
<hw_axi>	Associated hardware AXI core object.

Categories

Hardware

Description

Define a read or write transaction for the JTAG to AXI Master core, hw_axi object, specified by the `get_hw_axis` command.

The JTAG to AXI Master is a customizable IP core that works as an AXI Master to drive AXI transactions and drive AXI signals that are internal to the hardware device. The JTAG-AXI core supports all memory-mapped AXI interfaces, except AXI4-Stream, and supports the AXI-Lite protocol. Detailed documentation on the IP core can be found in the *LogiCORE IP JTAG to AXI Master Product Guide (PG174)*.

AXI transactions are read/write burst transactions from the JTAG to AXI Master core onto AXI signals connected to the core. The AXI transaction lets you configure aspects of the read or write transaction such as the data to send and the address to send it to. These defined transactions are stored as properties of the specified hw_axi object, waiting to be run and reported using the `run_hw_axi` and `report_hw_axi_txn` commands.

The command returns the name of the hw_axi_txn object created, or returns an error if it fails.

Arguments

`-address <arg>` - (Optional) Specify the address of the register on the hw_axi object to read from, or write into. Default address 0000.

`-data <arg>` - (Optional) The data value specified in hexadecimal format to write into the address location of the hw_axi for WRITE transactions. The default data value is all zeros.

`-type [ READ | WRITE ]` - (Required) Specify the AXI transaction to READ from the specified address, or WRITE into it.

`-len <arg>` - (Optional) The length of the READ or WRITE transaction, specified as the number of data words to read or write, based on the `-size` option. The default is 1.

`-burst <arg>` - (Optional) The type of AXI bursts to perform. Bursts can be specified as INCR, FIXED, or WRAP. The default data burst is incremental (INCR).

`-cache <arg>` - (Optional) The AXI command cache to implement, specified in decimal form. The default value is 3. For more information on read/write cache settings refer to the *LogiCORE IP Product Guide: JTAG to AXI Master (PG174)*.

`-id <arg>` - (Optional) The ID to assign to the AXI transaction, specified in decimal form. This lets the tool identify responses to various read and write transactions.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - (Required) The name to give to the newly created AXI transaction object. This name can be used to access or recall the transaction.

`<hw_axi>` - (Required) The hw_axi object to define the transaction for. The hw_axi must be specified as an object returned by the `get_hw_axi` command, and can not simply be specified by name.

Example

The following example specifies a WRITE transaction, with a data value of decimal 10 specified as a hexadecimal value, to be written in a 2 data word transaction for 32-bit data words:

```
create_hw_txn write_1 -type WRITE -data 0000_0000_0000_000A -size 32 -len 2 \
[get_hw_axis hw_axi_1]
```

The following example creates a new hardware AXI transaction object, hw_axi_txn, which writes the specified 128 bit data stream into the specified address of the hw_axi object. The new AXI transaction is named write_txn:

```
create_hw_axi_txn write_txn [get_hw_axis hw_axi_1] -type WRITE \
-len 4 -data {44444444_33333333_22222222_11111111}
```

This example creates AXI read and write transactions, runs the hw_axi, and reports on the results:

```
create_hw_axi_txn wr_txn [lindex [get_hw_axis] 0] -address 80000000 \
-data {11112222 33334444 55556666 77778888} -len 4 -type write
create_hw_axi_txn rd_txn [lindex [get_hw_axis] 0] -address 80000000 \
-len 4 -type read
run_hw_axi [get_hw_axi_txns wr_txn]
set wr_report [report_hw_axi_txn wr_txn -w 32]
puts $wr_report
run_hw_axi [get_hw_axi_txns rd_txn]
set rd_report [report_hw_axi_txn rd_txn -w 32]
puts $rd_report
close_hw_target;
disconnect_hw_server;
```

This example creates a sequence of READ type hw_axi transactions, and then runs them:

```
# Read registers
create_hw_axi_txn -address [format %08x [expr $baseaddr + \
    $MM2S_VDMACR_OFFSET]] -type read txn00 [get_hw_axis hw_axi_1]
create_hw_axi_txn -address [format %08x [expr $baseaddr + \
    $MM2S_VDMASR_OFFSET]] -type read txn01 [get_hw_axis hw_axi_1]
create_hw_axi_txn -address [format %08x [expr $baseaddr + \
    $MM2S_REG_INDEX_OFFSET]] -type read txn02 [get_hw_axis hw_axi_1]
create_hw_axi_txn -address [format %08x [expr $baseaddr + \
    $PARK_PTR_REG_OFFSET]] -type read txn03 [get_hw_axis hw_axi_1]
create_hw_axi_txn -address [format %08x [expr $baseaddr + \
    $VERSION_OFFSET]] -type read txn04 [get_hw_axis hw_axi_1]
create_hw_axi_txn -address [format %08x [expr $baseaddr + \
    $S2MM_VDMACR_OFFSET]] -type read txn05 [get_hw_axis hw_axi_1]
```

```
create_hw_axi_txn -address [format %08x [expr $baseaddr + \
    $S2MM_VDMASR_OFFSET]] -type read txn06 [get_hw_axis hw_axi_1]
create_hw_axi_txn -address [format %08x [expr $baseaddr + \
    $S2MM_VDMA_IRQ_OFFSET]] -type read txn07 [get_hw_axis hw_axi_1]
run_hw_axi -quiet [get_hw_axi_txns]
```

See Also

- delete_hw_axi_txn
- get_hw_axis
- get_hw_axi_txns
- refresh_hw_axi
- report_hw_axi_txn
- reset_hw_axi
- run_hw_axi

Send Feedback

create_hw_bitstream

Read bitstream file into memory

Syntax

```
create_hw_bitstream -hw_device <arg> [-mask <arg>] [-nky <arg>]
    [-detect_partial] [-quiet] [-verbose] [<file>]
```

Usage

Name	Description
-hw_device	Target hw_device connection
[-mask]	Mask file for hw device
[-nky]	Encryption file for hw device
[-detect_partial]	detects partial bitstream
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<file>]	Bitstream filename

Categories

Hardware

Description

Read a bitstream file, created with the `write_bitstream` command, to create a hw_bitstream object, and associate that object with a hw_device object in the Hardware Manager feature of the Vivado Design Suite.

The hw_bitstream object is associated with the specified hw_device through the PROGRAM.HW_BITSTREAM property on the device. This property is automatically set by the `create_hw_bitstream` command. The PROGRAM.FILE property is also set to reflect the file path of the specified bitstream file.

Note: A hw_bitstream object is also automatically created and associated with a hw_device object when you use the `program_hw_devices` command.

The mask file written with the bitstream file, using the `write_bitstream -mask` command, is associated through the MASK property on the hw_bitstream object. The encryption key file required for use with an encrypted bitstream is associated through the ENCRYPTION.FILE property on the hw_bitstream object. These files are associated with the hw_bitstream object using the `-mask` and `-nky` options.

The created hw_bitstream object can be removed with the `delete_hw_bitstream` command.

This command returns the name of the hw_bitstream object created, or returns an error if it fails.

Arguments

`-hw_device <arg>` - (Required) Specify the hw_device object to associate the hw_bitstream object with. The hw_device must be specified as an object as returned by the `get_hw_devices` or `current_hw_device` commands.

`-mask <arg>` - (Optional) Specify the mask file to use with the device indicating which bits in the bitstream should be compared to readback data for verification purposes. The mask file is written by the `write_bitstream -mask_file` command, and is used by the `verify_hw_bitstream` command to verify the programmed bitstream is correct. The specified mask is defined with the MASK property on the created hw_bitstream object.

`-nky <arg>` - (Optional) Specify the encryption key file to program into the eFUSE registers or battery backed-up SRAM (BBR). The encryption key defined in the NKY or NKZ file is written by the `write_bitstream` command, and is required for use with an encrypted bitstream. The specified encryption key file is defined with the ENCRYPTION.FILE property on the created hw_bitstream object. The encryption key value is extracted from the file and defined on the ENCRYPTION.KEY property.

`-detect_partial` - (Optional) Detects a partial bitstream file created by the `write_bitstream -reference_bitfile` command. This enables incremental programming of the hardware device with the partial bitstream file.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file>` - (Required) The name of the bitstream file to read.

Note: If the path is not specified as part of the file name, the tool will search for the specified file in the current working directory and then in the directory from which the tool was launched.

Example

The following example creates a hw_bitstream object from the specified bitstream file, and associates it with the current hw_device object:

```
create_hw_bitstream -hw_device [current_hw_device] C:/Data/design1.bit
```

The following example creates a hw_bitstream object for the current hw_device, and specifies the mask file and encryption key file (nky) to associate with the bitstream:

```
create_hw_bitstream -hw_device [current_hw_device] \
  -mask ./project_cpu_encrypt.runs/impl_1/top.msk \
  -nky ./project_cpu_encrypt.runs/impl_1/top.nky \
  ./project_cpu_encrypt.runs/impl_1/top.bit
```

See Also

- current_hw_device
- delete_hw_bitstream
- get_hw_devices
- program_hw_devices
- set_property
- write_bitstream

Send Feedback

create_hw_cfgmem

Read cfgmem file into memory

Syntax

```
create_hw_cfgmem -hw_device <arg> [-quiet] [-verbose] <mem_device>
```

Usage

Name	Description
-hw_device	hw_device object with which to associate hw_cfgmem object
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<mem_device>	name of flash memory device as returned by get_cfgmem_parts

Categories

Hardware

Description

Create a hw_cfgmem object associated with the specified hw_device.

The process whereby the bitstream data is loaded or programmed into the Xilinx® FPGA is called configuration. Configuration is designed to be flexible to accommodate different application needs and, wherever possible, to leverage existing system resources to minimize system costs.

Xilinx FPGAs are configured by loading design-specific configuration data, in the form of a bitstream file, into the internal memory of the hw_device. The hw_cfgmem defines a flash memory device used for configuring and booting the Xilinx FPGA device. Once the hw_cfgmem object is created, and associated with the hw_device, the configuration memory can be programmed with the bitstream and other data using the `program_hw_cfgmem` command.

The hw_cfgmem object is associated with the specified hw_device object through the PROGRAM.HW_CFGMEM property on the device object. Use the `get_hw_cfgmems` command to work with the hw_cfgmem object, or use the `get_property` command to obtain the object from the hw_device:

```
get_property PROGRAM.HW_CFGMEM [current_hw_device]
```

> 💡 **TIP:** *When creating a new hw_cfgmem object, you can also associate the object with a Tcl variable as shown in the example below. By referring to the variable, you can set properties on the object, and use the object with other Tcl commands like* `program_hw_cfgmem` *or* `readback_hw_cfgmem`.

Send Feedback

This command returns the created hw_cfgmem object, or returns an error if it fails.

Arguments

`-hw_device <arg>` - (Required) Specify the hw_device object to associate the memory configuration device with. The hw_device must be specified as an object as returned by the `get_hw_devices` command or `current_hw_device` command.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<mem_device> - (Required) Specify the flash memory device to use for configuring the associated hw_device. The configuration memory must be specified as a cfgmem_part object object using the `get_cfgmem_parts` command.

Note: The COMPATIBLE_PARTS property of the cfgmem_part object identifies which Xilinx devices a configuration memory is compatible with.

Example

The following example:

1. Gets the PART associated with the `current_hw_device`.

2. Gets a list of cfgmem_parts that are compatible with the device part.

3. Uses one of the compatible parts to create a new hw_cfgmem object.

4. Sets the PROGRAM.FILE property of the current hw_cfgmem object to the cfgmem file created with the `write_cfgmem` command.

5. Programs the current hw_cfgmem object.

```
set devPart [get_property PART [current_hw_device]]
set cfgParts [get_cfgmem_parts -of [get_parts $devPart]]
create_hw_cfgmem -hw_device [current_hw_device] [lindex $cfgParts 0 ]
set_property PROGRAM.FILE {C:/Data/cfgmem_file.mcs} [current_hw_cfgmem]
program_hw_cfgmem [current_hw_cfgmem]
```

See Also

- current_hw_cfgmem
- current_hw_device

Send Feedback

- delete_hw_cfgmem
- get_cfgmem_parts
- get_hw_cfgmems
- get_property
- program_hw_cfgmem
- readback_hw_cfgmem
- write_cfgmem

create_hw_device

Create a hw_device (jtag chain) on an open target

Syntax

```
create_hw_device [-idcode <arg>] [-idcode2 <arg>] [-irlength <arg>]
     [-mask <arg>] [-part <arg>] [-quiet] [-verbose]
```

Usage

Name	Description
[-idcode]	hexadecimal device id code
[-idcode2]	hexadecimal device id code
[-irlength]	decimal device ir length
[-mask]	hexadecimal device mask value
[-part]	part type of device to create
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Hardware

Description

The Vivado hardware manager supports programming of hardware devices through the use of Serial Vector Format (SVF) files. SVF files are ASCII files that contain both programming instructions and configuration data. These files are used by ATE machines and embedded controllers to perform boundary-scan operations. The SVF file captures the JTAG commands needed to program the bitstream directly into a Xilinx device, or indirectly into a flash memory device. The SVF file can be written using the `write_hw_svf` command, or applied to an open hw_target through the `execute_hw_svf` command. Refer to the *Vivado Design Suite User Guide: Programming and Debugging* (UG908) for more information.

The specific process for creating the hw_svf file is:

1. Create an SVF target using `create_hw_target`.

2. Open the SVF target.

3. Create one or more devices on the SVF target using `create_hw_device`.

4. Program the devices using commands like `program_hw_devices`.

5. Write the SVF file of operation commands using `write_hw_svf`.

The `create_hw_device` command creates a hw_device object on an open SVF target, adding it to the JTAG chain. This device can be queried and programmed like other hw_targets using commands like `get_hw_devices` and `program_hw_devices`.

You can create both Xilinx devices and user-defined parts to add to the JTAG chain on the open SVF hw_target. For Xilinx devices, simply specify a recognized part number and the Vivado tool will define it with the appropriate details. For user-defined parts you must specify the JTAG ID code, IR length, and mask details using the appropriate options. User-defined parts are added as space-holder devices to the JTAG chain as on the SVF hw_target. You can get the user-part with `get_hw_devices` command, and query the properties of the part with `report_property`, but you cannot program user-parts.

> ⭐ **IMPORTANT!** *You should create all the devices to define the JTAG chain for the SVF target, before performing any operations on the JTAG chain. If you mix* `create_hw_device` *commands with programming commands the JTAG chain referenced in the SVF file will be improperly defined and will not work during* `execute_hw_svf`.

After creating the hw_device on the SVF target, you can exercise the device with supported operations such as associating a bitstream file (.bit) and programming the device:

```
set_property PROGRAM.FILE {C:/Data/design.bit} [current_hw_device]
program_hw_devices [current_hw_device]
```

The `create_hw_device` command returns nothing if successful, and returns an error if it fails.

Arguments

`-idcode <arg>` - (Optional) Specifies the JTAG ID code for the user-defined device. This is not required when specifying a Xilinx part.

`-irlength <arg>` - (Optional) Specifies the JTAG Instruction Register (IR) length for the user-defined device. This is not required when specifying a Xilinx part.

`-mask <arg>` - (Optional) Specifies the mask used to read the Data Registers (DR) from the device in the JTAG chain. The mask defines which bits in the device response are valid. Masked bits will be ignored. This is not required when specifying a Xilinx part.

`-part <arg>` - Specifies a Xilinx part for the device, or specifies a user-defined part name.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example creates an SVF target, opens that target, and creates a new hw_device on the target:

```
create_hw_target my_svf_target
open_hw_target
create_hw_device -part xc7k325t
```

This example demonstrates the correct order of creating multiple devices on an SVF target. An SVF target is created and opened, then a Xilinx device, a user part, and a second Xilinx device are created on the current target. The bitstream properties are defined for the two Xilinx devices, the devices are programmed, and an SVF file is written:

```
open_hw
connect_hw_server
create_hw_target my_svf_target
open_hw_target
create_hw_device -part xc7k325t
create_hw_device -idcode 01234567 -irlength 8 -mask ffffffff -part userPart1
create_hw_device -part xcku9p
set_property PROGRAM.FILE {C:/Data/k7_design.bit} [lindex [get_hw_devices] 0]
set_property PROGRAM.FILE {C:/Data/ku_design.bit} [lindex [get_hw_devices] 2]
program_hw_devices [lindex [get_hw_devices] 0]
program_hw_devices [lindex [get_hw_devices] 2]
write_hw_svf C:/Data/myDesign.svf
```

See Also

- create_hw_target
- current_hw_device
- current_hw_target
- get_hw_devices
- get_hw_targets
- open_hw_target
- program_hw_devices
- set_property

create_hw_probe

Create hardware probe object

Syntax

```
create_hw_probe [-no_gui_update] [-map <arg>] [-quiet] [-verbose] <name>
    <core>
```

Returns

New hardware probe object.

Usage

Name	Description
[-no_gui_update]	Defer GUI update.
[-map]	Declaration of bits. Default: 0
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<name>	Name of new object. Bus probes have range appended.
<core>	Associated hardware ILA core object.

Categories

Hardware

Description

This command creates a new user-defined probe on the specified ILA core to define triggers and view data in the Vivado Logic Analyzer. The new probe can combine specific bit values of existing probes to simplify or clarify the data presented in the waveform viewer. Captured data samples from the user-defined probe can be reported with the `list_hw_samples` command.

User-defined probes can map bit values from a single physical probe on the ILA core, or can combine bit values from multiple physical probes onto a single user-defined probe. Probes that map bits from a single probe can be used to create triggers and view data. Probes that combine bits from multiple physical probes can only be used for viewing data in the Vivado Logic Analyzer.

You can delete user-defined probes with the `delete_hw_probe` command.

The `create_hw_probe` command returns the user-defined probe name when successful, or returns an error if it fails.

Send Feedback

Arguments

`-no_gui_update` - (Optional) Do not update the GUI in the Vivado logic analyzer to reflect the addition of the user-defined probe.

`-map <arg>` - (Optional) Specifies the physical probe port name and signal bits to map into the new user-defined probe. Physical probe ports are the ports on the specified hw_ila object, and are related to signals being probed by the ILA core. The `-map` argument is specified as a list of physical probe port names and bits:

```
-map {0011 probe3[19] probe3[6]}
```

💡 **TIP:** *The map can also include constant values as indicated above.*

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - Specifies the name to assign to the user-defined probe, as a combination of name and probe width, probeName[0:3] for instance. If no width is specified, the default probe width is one bit.

Note: The user-defined probe width must match the number of bits specified in the `-map` argument.

<core> - Specifies the ILA core object. The hw_ila can be specified by name or as an object returned by the `get_hw_ilas` command.

Examples

The following example maps bits from multiple physical probes onto a new user-defined probe on the specified ILA core:

```
create_hw_probe -map {0011 probe5[3:0] probe8 probe9} myProbeAR[9:0]
hw_ila_1
```

💡 **TIP:** *The `-map` option combines 10 bits onto the new probe, so the probe name specifies a matching port width.*

The following example creates a hw_probe with copies of the most-significant bit to sign-extend a 30-bit signal to align it with other 32-bit signed signals:

```
create_hw_probe -map {probe0[29] probe0[29] probe0[29:0]} \
mySignExtendedProbe[31:0] [get_hw_ilas hw_ila_1]
```

See Also

- delete_hw_probe
- get_hw_ilas
- get_hw_probes

create_hw_sio_link

Create a new link between hardware RX and TX endpoints. There must be at least one hardware TX or RX endpoint specified. If one is missing, the endpoint will be treated as Unknown. The unknown endpoint can be renamed in a link property.

Syntax

```
create_hw_sio_link [-description <arg>] [-quiet] [-verbose] [<hw_sio_rx>]
    [<hw_sio_tx>]
```

Returns

The new hardware SIO link.

Usage

Name	Description
[-description]	Description of link. Default: Link object name
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<hw_sio_rx>]	RX endpoint. Default: None
[<hw_sio_tx>]	TX endpoint. Default: None

Categories

Hardware

Description

Define a communication links between transmitter (TX) and receiver (RX) objects on the GTs of the IBERT debug core implemented on the current hardware device.

Vivado Serial I/O analyzer is a link-based analyzer, which lets you link between any transmitter and receiver within the IBERT design. The links define the communication paths and protocols between transmitters and receivers of the GigaBit transceivers on the device. You can configure the links by using the `set_property` command to specify property values on the link object. Refer to the *Vivado Design Suite User Guide: Programming and Debugging (UG908)* for more information on configuring links.

This command returns the created hw_sio_link object, or returns an error if it fails.

Send Feedback

Arguments

`-description <arg>` - (Optional) Provide a brief description that acts as a label for the link. This description is stored on the DESCRIPTION property of the hw_sio_link object.

> **TIP:** *The NAME property of the hw_sio_link objects is a full path designation of the link from the transmitter (TX) to receiver (RX) GTs on the IBERT debug core, including the hw_server, hw_target, hw_device, and hw_sio_ibert objects. Any description you provide here can be used as a shortcut to help locate the link.*

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<hw_sio_rx>` - (Optional) Specify the receiver element of the GT to include in the communication link. The receiver must be specified as an object as returned by the `get_hw_rxs` command.

`<hw_sio_tx>` - (Optional) Specify the transmitter element of the GT to include in the communication link. The transmitter must be specified as an object as returned by the `get_hw_txs` command.

> **IMPORTANT!** *Although the receiver and transmitter are marked as optional, at least one must be provided to create a link. A link with only a transmitter or receiver is considered an open-ended link.*

Example

The following example creates a communication link between the specifed RX and TX

```
create_hw_sio_link -description Link_12  [get_hw_sio_txs *MGT_X0Y12*] \
[get_hw_sio_rxs *MGT_X0Y12*]
```

> **TIP:** *In the example above the TX precedes the RX. The order of these objects is not significant.*

See Also

- create_hw_sio_linkgroup
- current_hw_device
- get_hw_sio_iberts
- get_hw_sio_links

Send Feedback

- get_hw_sio_linkgroups
- remove_hw_sio_link
- remove_hw_sio_linkgroup

create_hw_sio_linkgroup

Create a new hardware SIO link group.

Syntax

```
create_hw_sio_linkgroup [-description <arg>] [-quiet] [-verbose]
    <hw_sio_links>
```

Returns

The new hardware SIO link group.

Usage

Name	Description
[-description]	Description of link group. Default: Link group object name
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hw_sio_links>	hardware SIO links

Categories

[Hardware](#)

Description

Create a new group to associate the specified TX to RX communication links on the IBERT debug core implemented on the current device.

Vivado Serial I/O analyzer is a link-based analyzer. The links define the communication paths and protocols between transmitters and receivers of the GigaBit transceivers on the device. Link groups, or hw_sio_linkgroup objects, let you associate links into related groups, to collectively configure properties and run scans.

This command returns the name of the linkgroup created, or returns an error if the command fails.

Arguments

-description <arg> - (Optional) Provide a brief description that acts as a label for the link group. This description is stored on the DESCRIPTION property of the hw_sio_linkgroup object.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Send Feedback

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_sio_links> - (Required) Specify one or more hw_sio_link objects to associate together as a group. The hw_sio_links must be specified as objects as returned by the `create_hw_sio_link` or `get_hw_sio_links` commands.

Note: If you include a hw_sio_link object that already belongs to another linkgroup, the command will return an error.

Example

The following example uses the DESCRIPTION property on hw_sio_link objects to identify which links to add to a new link group:

```
create_hw_sio_linkgroup -description "LoopBack Links" \
    [get_hw_sio_links -filter {DESCRIPTION == Link_12 || DESCRIPTION == \
    "Link 9" || DESCRIPTION == "Link 10" || DESCRIPTION == "Link 11" }]
```

See Also

- create_hw_sio_link
- current_hw_device
- get_hw_sio_iberts
- get_hw_sio_links
- get_hw_sio_linkgroups
- remove_hw_sio_link
- remove_hw_sio_linkgroup

Send Feedback

create_hw_sio_scan

Create a new hardware SIO scan. If a Link object is passed in, it must have a RX Endpoint object.

Syntax

```
create_hw_sio_scan [-description <arg>] [-link_settings <arg>] [-quiet]
    [-verbose] <scan_type> <hw_sio_object>
```

Returns

The new hardware SIO scan.

Usage

Name	Description
[-description]	Description of scan Default: Scan object name
[-link_settings]	List of Link properties and values to set before running the scan. Default: None
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<scan_type>	Scan Type Options: 1d_bathtub, 2d_full_eye
<hw_sio_object>	RX endpoint or Link object to perform scan on.

Categories

[Hardware](#)

Description

Create a serial I/O analyzer scan object for the specified communication link on the IBERT debug core.

To analyze the margin of a given link, it is often helpful to run a scan of the link using the specialized Eye Scan hardware of Xilinx UltraScale devices or 7 series FPGAs. The Vivado serial I/O analyzer feature lets you to create, run, and save link scans.

This command creates and returns a link scan object that you can use with the `run_hw_sio_scan` command to run analysis on the specified links, or GT receivers. You can also save the scan to disk using the `write_hw_sio_scan` command.

You can remove the created scan object using `remove_hw_sio_scan`.

This command returns the hw_sio_scan object, or returns an error if the command fails.

Send Feedback

Arguments

`-description <arg>` - (Optional) Provide a brief description that acts as a label for the serial I/O analyzer scan. The description can be used to identify the <hw_sio_scan> object. For instance, you can identify the receiver port, so that when you are sweeping many ports you can keep track of which port the scan plot s for.

`-link_settings <arg>` - (Optional) Specify a list of Link properties and values to set before running the scan. If no link settings are provided, the default settings are used. Refer to *Vivado Design Suite User Guide: Programming and Debugging (UG908)* for a description of scan properties and settings.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<scan_type> - (Required) Specify the scan type. Valid types include:

- 1d_bathtub - Scan all horizontal sampling points through the 0 vertical axis.

- 2d_full_eye - Scan all horizontal and vertical sampling points to create an "eye".

<hw_sio_object> - (Required) Specify the IBERT debug core link, hw_sio_link, or receiver, hw_sio_rx, to define the scan object for. The link or receiver must be specified as objects as returned by the `get_hw_sio_links` or `get_hw_sio_rxs` commands.

Note: The create_hw_sio_scan command requires the hw_sio_object to be specified as a list of one object.

Example

The following example defines a scan for the specified link:

```
set xil_newScan [create_hw_sio_scan -description {LoopBack} 2d_full_eye \
[lindex [get_hw_sio_links *MGT_X0Y10/TX*] 0 ]]
run_hw_sio_scan [get_hw_sio_scans $xil_newScan]
```

See Also

- create_hw_sio_sweep
- current_hw_device
- get_hw_sio_scans
- get_hw_sio_sweeps

Send Feedback

- remove_hw_sio_scan
- run_hw_sio_scan
- stop_hw_sio_scan
- wait_on_hw_sio_scan
- write_hw_sio_scan

create_hw_sio_sweep

Create a new hardware SIO sweep. If a Link object is passed in, it must have a RX Endpoint object.

Syntax

```
create_hw_sio_sweep [-description <arg>] [-iteration_settings <arg>]
    [-quiet] [-verbose] <scan_type> [<hw_sio_link>]
```

Returns

The new hardware SIO sweep.

Usage

Name	Description
[-description]	Description of sweep Default: Sweep object name
[-iteration_settings]	List of LINK_SETTINGS for each scan to set before running the sweep. Default: None
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<scan_type>	Sweep Type Options: 1d_bathtub, 2d_full_eye
[<hw_sio_link>]	Link object to perform sweep on. Default: None

Categories

Hardware

Description

Create a serial I/O analyzer link sweep object to run multiple scans across a range of values.

To analyze the margin of a given link, it is often helpful to run a scan of the link using the specialized features of Xilinx® UltraScale™ devices or 7 series FPGAs. It can also be helpful to run multiple scans on a the link with different configuration settings for the GTs. This can help you determine which settings are best for your design. The Vivado® serial I/O analyzer feature enables you to define, run, and save link sweeps, or collections of link scans run across a range of values.

This command creates and returns a link sweep object that you can use with the `run_hw_sio_sweep` command to run analysis on the specified links, or GT receivers. You can also save the sweep scan to disk using the `write_hw_sio_sweep` command.

You can remove the created sweep object using `remove_hw_sio_sweep`.

Send Feedback

This command returns the hw_sio_sweep object, or returns an error if the command fails.

Arguments

`-description <arg>` - (Optional) Provide a brief description that acts as a label for the serial I/O analyzer sweep scan.

`-iteration_settings <arg>` - (Optional) Specify a list of properties to vary across multiple scans. Refer to *Vivado Design Suite User Guide: Programming and Debugging (UG908)* for a description of iteration settings.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<scan_type>` - (Required) Specify the scan type. Valid types include:

- 1d_bathtub - Scan all horizontal sampling points through the 0 vertical axis.

- 2d_full_eye - Scan all horizontal and vertical sampling points to create an "eye".

> 💡 **TIP:** *The results of the bathtub scan can be saved to a file with write_hw__sio_scan, but the plot cannot be displayed in the Vivado serial I/O analyzer using display_hw_sio_scan.*

`<hw_sio_link>` - (Optional) Specify the hw_sio_link to define the sweep object for. The link must be specified as objects as returned by the `get_hw_sio_links` command.

Example

The following example defines a variable for the hw_sio_sweep object created, then runs the sweep scan:

```
set xil_newSweep [create_hw_sio_sweep -description {Sweep 0} 2d_full_eye \
    [lindex [get_hw_sio_links *MGT_X0Y10/TX*] 0 ]]
run_hw_sio_sweep [get_hw_sio_sweeps $xil_newSweep]
```

See Also

- create_hw_sio_scan
- current_hw_device
- get_hw_sio_scans

- get_hw_sio_sweeps
- remove_hw_sio_scan
- remove_hw_sio_sweep
- run_hw_sio_sweep
- stop_hw_sio_sweep
- wait_on_hw_sio_sweep
- write_hw_sio_sweep

create_hw_target

Create a hw_target (jtag chain) and set its name

Syntax

```
create_hw_target [-copy <arg>] [-quiet] [-verbose] <target_name>
```

Returns

Hardware targets

Usage

Name	Description
`[-copy]`	hardware target copy Default: copy of existing target
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<target_name>`	name of hardware target to create

Categories

Hardware

Description

The Vivado hardware manager supports programming of hardware devices through the use of Serial Vector Format (SVF) files. SVF files are ASCII files that contain both programming instructions and configuration data. These files are used by ATE machines and embedded controllers to perform boundary-scan operations. The SVF file captures the JTAG commands needed to program the bitstream directly into a Xilinx device, or indirectly into a flash memory device. The SVF file can be written using the `write_hw_svf` command, or applied to an open hw_target through the `execute_hw_svf` command. Refer to the *Vivado Design Suite User Guide: Programming and Debugging* (UG908) for more information.

The specific process for creating the hw_svf file is:

1. Create an SVF target using `create_hw_target`.

2. Open the SVF target.

3. Create one or more devices on the SVF target using `create_hw_device`.

4. Program the devices using commands like `program_hw_devices`.

5. Write the SVF file of operation commands using `write_hw_svf`.

The `create_hw_target` command creates an SVF hw_target object on the current hw_server that can be used as a platform for programming devices, and exporting the programming commands in an SVF file. The SVF target, is a hw_target object that can be queried and managed like other hw_targets using commands like `get_hw_targets` and `current_hw_target`.

Note: When using the SVF flow, Xilinx recommends that you connect to a local hw_server on your system, as the SVF target does not require connection to an actual hardware board or device.

SVF hw_targets can be identified by the boolean IS_SVF property that can be returned by `get_property` or `report_property` commands. This property is TRUE for SVF targets.

This command returns nothing if successful, or returns an error if it fails.

Arguments

`-copy <arg>` - (Optional) Specifies that the new SVF hw_target should be a copy of an existing hw_target. The argument specifies a physical hw_target or SVF hw_target as returned by the `get_hw_targets` command.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<target_name> - Specifies the name to assign to the new SVF hw_target object.

Examples

The following example creates a SVF hw_target object that is a copy of the specified hw_target:

```
create_hw_target -copy [get_hw_targets *210203327996A] svfTarget
```

The following example gets the currently defined SVF hw_target objects:

```
get_hw_targets -filter {IS_SVF}
```

The following example shows all of the steps needed for the SVF flow. First open the Vivado hardware manager and connect to a local hw_server; create and open an SVF hw_target; add a hw_device and program the bitstream into this device; and write the SVF file capturing the programming commands for the device:

```
open_hw
connect_hw_server
create_hw_target my_svf_target
open_hw_target
create_hw_device -part xc7k325t
set_property PROGRAM.FILE {C:/Data/k7_design.bit} [current_hw_device]
program_hw_devices [current_hw_device]
write_hw_svf my_xc7k325t.svf
close_hw_target
```

See Also

- connect_hw_server
- create_hw_device
- current_hw_target
- get_hw_targets
- get_property
- program_hw_devices
- report_hw_targets
- report_property
- set_property
- write_hw_svf

Send Feedback

create_interface

Create a new I/O port interface

Syntax

```
create_interface [-parent <arg>] [-quiet] [-verbose] <name>
```

Returns

New interface object.

Usage

Name	Description
[-parent]	Assign new interface to this parent interface
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<name>	Name for new I/O port interface

Categories

[PinPlanning](#)

Description

Creates a new interface for grouping scalar or differential I/O ports.

Arguments

`-parent <arg>` - (Optional) Assign the new interface to the specified parent interface.

Note: If the specified parent interface does not exist, an error will be returned.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - (Required) The name of the I/O port interface to create.

Send Feedback

Examples

Create a new USB interface:

```
create_interface USB0
```

Create an Ethernet interface within the specified parent interface:

```
create_interface -parent Top_Int ENET0
```

See Also

- delete_interface
- create_port
- make_diff_pair_ports
- place_ports
- remove_port
- set_package_pin_val
- split_diff_pair_ports

Send Feedback

create_ip

Create an instance of a configurable IP and add it to the fileset

Syntax

```
create_ip [-vlnv <arg>] -module_name <arg> [-dir <arg>] [-force]
    [-allow_hidden] [-vendor <arg>] [-library <arg>] [-name <arg>]
    [-version <arg>] [-revision <arg>] [-quiet] [-verbose]
```

Returns

List of file objects that were added.

Usage

Name	Description
[-vlnv]	VLNV string for the Catalog IP from which the new IP will be created (colon delimited Vendor, Library, Name, Version)
-module_name	Name for the new IP that will be added to the project
[-dir]	Directory path for remote IP to be created and managed outside the project
[-force]	Overwrite existing IP instance; allowed only with -dir option
[-allow_hidden]	Allow hidden cores to be instantiated
[-vendor]	IP Vendor name
[-library]	IP Library name
[-name]	IP Name
[-version]	IP Version
[-revision]	(Optional) IP core revision
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

IPFlow

Description

This command creates an XCI file for a configurable IP core from the IP catalog, and adds it to the source files of the current project. This creates an IP source object which must be instantiated into the HDL design to create an instance of the IP core in the netlist.

For multiple instances of the same core, simply instantiate the core module into the HDL design as many times as needed. However, to use the same IP core with different customizations, use the `create_ip` command to create separate IP source objects.

Send Feedback

The `create_ip` command is used to import IP cores from the current IP catalog. Use the `import_ip` command to read existing XCI and XCO files directly, without having to add IP to a catalog.

This command returns a transcript of the IP generation process, concluding with the file path and name of the imported IP core file.

Note: IP cores are native to Vivado, and can be customized and regenerated within that tool. The `convert_ip` command lets you to convert legacy IP to native IP supported by Vivado.

Arguments

`-vlnv <arg>` - (Optional) Specifies the VLNV string for the existing Catalog IP from which the new IP will be created. The VLNV is the *Vendor:Library:Name:Version* string which identifies the IP in the catalog. The VLNV string maps to the IPDEF property on the IP core.

Note: You must specify either `-vlnv` or all of `-vendor`, `-library`, `-name`, and `-version`.

`-module_name <arg>` - (Required) Specifies the name for the new IP instance that will be created. The module is created with the `<module_name>/<module_name>.xci` naming convention.

`-dir <arg>` - (Optional) The directory to write the IP core files into. If this option is not specified, the IP files (`.xci`, `.veo`...) are written in the hierarchy of the `<project_name>.srcs` directory.

`-force` - (Optional) Overwrite an existing IP instance of the same <module_name>, if one exists in the specified directory. All files associated with the existing IP will be overwritten by files associated with the newly created IP. This option can only be used with the `-dir` option.

`-vendor <arg>` - (Optional) Specifies the vendor name for the IP's creator.

`-library <arg>` - (Optional) Specifies the IP library from which the core should be added.

`-name <arg>` - (Optional) Specifies the name of the IP core in the catalog.

`-version <arg>` - (Optional) Specifies the version number for the IP core.

Note: You must specify either `-vlnv` or all of `-vendor`, `-library`, `-name`, and `-version`.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The example below imports the IP core specified by the `-vlnv` string, and gives it the specified module name in the current project:

```
create_ip -vlnv xilinx.com:ip:c_addsub:11.0 -module_name test_addr
```

The following example, from Vivado, creates an IP block with the specified `-vendor`, `-library`, `-name`, `-version` values, and assigns it the specified module name. After the IP is created, attributes of the IP are customized using `set_property` commands. Then the instantiation template and the synthesis targets are generated for the IP:

```
create_ip -name c_addsub -version 11.0 -vendor xilinx.com -library ip \
    -module_name c_addsub_v11_0_0
set_property CONFIG.COMPONENT_NAME  c_addsub_v11_0_0 \
    [get_ips c_addsub_v11_0_0]
set_property CONFIG.A_WIDTH  32 [get_ips c_addsub_v11_0_0]
set_property CONFIG.B_WIDTH 32 [get_ips c_addsub_v11_0_0]
set_property CONFIG.ADD_MODE Add_Subtract [get_ips c_addsub_v11_0_0]
set_property CONFIG.C_IN true [get_ips c_addsub_v11_0_0]
generate_target  {instantiation_template synthesis} \
    [get_files C:/Data/c_addsub_v11_0_0/c_addsub_v11_0_0.xci \
    -of_objects [get_filesets sources_1]]
```

See Also

- convert_ips
- generate_target
- import_ip
- upgrade_ip
- validate_ip

Send Feedback

create_ip_run

Creates a run for the given IP.

Syntax

```
create_ip_run [-force] [-quiet] [-verbose] <objects>
```

Usage

Name	Description
[-force]	Force regeneration of products of the given IP.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<objects>	All of the IP objects (from get_ips or get_files) for which a run needs to be generated for.

Categories

[Project](#), [IPFlow](#)

Description

Defines a synthesis and implementation run for a single IP object as specified by the `get_ips` command, or for the specified IP core file (XCI) as specified by the `get_files` command.

The IP run is used to generate the synthesis design checkpoint file (DCP) to support the out-of-context (OOC) IP flow, or to synthesize and implement an IP module in the OOC hierarchical design flow.

Two runs are created: one for synthesis, and one for implementation. The runs are named after the IP core and the run type, `<ip_name>_synth_1` and `<ip_name>_impl_1`.

The IP source files required to synthesize the run are copied into the IP run directory. The attributes of the run can be configured with the use of the `set_property` command.

This command returns the name of the synthesis run created for the IP module.

Arguments

`-force` - (Optional) Force the creation of an IP run, even if the generated output products for the specified IP are all current.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<object>` - (Required) An IP object as returned by the `get_ips` command, or an IP file (XCI) as specified by the `get_files` command.

> **TIP:** *Only a single IP may be specified.*

Examples

The following example creates synthesis and implementation runs for the specified IP module:

```
create_ip_run [get_ips add1]
```

See Also

- create_run
- get_files
- get_ips
- set_property

create_macro

Create a Macro

Syntax

```
create_macro [-quiet] [-verbose] <name>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<name>	Macro to create.

Categories

[XDC](XDC)

Description

Create a macro for the relative placement of cells.

Macros are primarily used to place small groups of associated cells together to improve resource efficiency and enable faster interconnections. The `create_macro` command lets you define macros in an open synthesized or implemented design for relative placement by `place_design`, like RPMs defined by the RLOC constraint in RTL source files. Refer to the *Vivado Design Suite User Guide: Implementation (UG904)* for more information on defining relatively placed macros.

After creating the macro, specific cells can be assigned to the macro using the `update_macro` command. To change a currently defined macro, you must delete the macro with `delete_macro`, recreate the macro, and update the macro with the new contents. You cannot simply overwrite an existing macro.

Use `delete_macro` to delete a defined macro. Use `get_macros` to return a list of currently defined macros in the design.

This command operates silently and does not return anything.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Send Feedback

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - (Required) Specify the name of the macro to create.

Examples

The following example creates a macro called usbMacro1:

```
create_macro usbMacro1
```

See Also

- delete_macros
- get_macros
- place_design
- update_macro

create_net

Create nets in the current design

Syntax

```
create_net [-from <arg>] [-to <arg>] [-quiet] [-verbose] <nets>...
```

Usage

Name	Description
[-from]	Starting bus index
[-to]	Ending bus index
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<nets>	Names of nets to create

Categories

Netlist

Description

Create new nets in the current netlist of an open Synthesized or Implemented Design.

Note: You cannot add nets to library macros, or macro-primitives.

Nets can be created hierarchically from the top-level of the design, or within any level of the hierarchy by specifying the hierarchical net name.

Bus nets can be created with increasing or decreasing bus indexes, using negative and positive index values.

New nets are unconnected in the netlist at the time of creation. You must connect nets as desired using the `connect_net` command. Connected nets can be unconnected using the `disconnect_net` command, and can be removed from the netlist using the `remove_net` command.

Netlist editing changes the in-memory view of the netlist in the current design. It does not change the files in the source fileset, or change the persistent design on the disk. Changes made to the netlist may be saved to a design checkpoint using the `write_checkpoint` command, or may be exported to a netlist file such as Verilog, VHDL, or EDIF, using the appropriate `write_*` command.

Note: Netlist editing is not allowed on the elaborated RTL design.

Send Feedback

Arguments

`-from <arg>` - (Optional) The starting index of a new bus.

`-to <arg>` - (Optional) The ending index of a new bus.

Note: Specifying `-from` or `-to` without the other will results in a one-bit bus with index value specified by the `-from` or `-to` argument.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<nets>` - (Required) The names of nets to create. Net names can be specified from the top-level, as name only (net1), or can be specified within the design hierarchy by specifying the hierarchical net name (cell1/cellA/net1).

Example

The following example creates a new 24-bit bus in the current Synthesized or Implemented Design:

```
create_net tempBus -from 23 -to 0
```

See Also

- connect_net
- create_pin
- create_port
- disconnect_net
- get_nets
- remove_net
- resize_net_bus
- write_checkpoint
- write_edif
- write_verilog
- write_vhdl

create_partition_def

Create new PartitionDef

Syntax

```
create_partition_def -name <arg> -module <arg> [-library <arg>] [-quiet]
    [-verbose]
```

Usage

Name	Description
-name	Name of the PartitionDef
-module	Module name of the PartitionDef
[-library]	Library name of the module of PartitionDef
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Object, Partition

Description

> ⭐ **IMPORTANT!** *You must first define the project as a Partial Reconfiguration (PR) project by setting the PR_FLOW property on the project to TRUE, or by using the **Tools → Enable Partial Reconfiguration** command.*

The Partial Reconfiguration flow lets you create Partition Definitions (partitionDefs) from hierarchical cells in a design, and to specify reconfigurable modules (RMs) to be assigned to these partitionDefs to create a unique configurations of the design based on the combination of the core design and one or more RMs. The PR design flow requires the implementation of each PR configuration, resulting in partial bitstreams for the RMs, but complete bitstreams for each integrated configuration. Refer to the *Vivado Design Suite User Guide: Dynamic Function eXchange* (UG909) for more information.

The `create_partition_def` command defines a partitionDef object in a PR project from a specified hierarchical cell. The partitionDef defines a partition hierarchy that RMs can be assigned to for a specific PR configuration.

This command returns the name of the newly created partitionDef, or returns an error if the command fails.

Arguments

`-name <arg>` - (Required) The name to assign to the new partitionDef.

`-module <arg>` - (Required) The name of the hierarchical cell or module in the current project that defines the partitionDef. The hierarchical cell can be specified by name. The module specifies the hierarchical boundary of the partition definition. Reconfigurable modules (RMs) will be defined to fit within the hierarchical boundary of the partitionDef.

`-library <arg>` - (Optional) Specifies the library name to assign to the partitionDef. If no library is specified, the `xil_defaultlib` is used.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Example

The following example sets the PR_FLOW property on the current project, defines a new partitionDef and assigns the specified module, and then creates the reconfigurable module from the specified hierarchy:

```
set_property PR_FLOW 1 [current_project]
create_partition_def -name partDef1 -module fftTop \
-library xil_defaultlib
create_reconfig_module -name fftTop -define_from fftTop \
-partition_def [get_partition_defs partDef1]
```

See Also

- create_pr_configuration
- create_reconfig_module
- delete_partition_defs
- get_partition_defs
- set_property

Send Feedback

create_pblock

Create a new Pblock

Syntax

```
create_pblock [-quiet] [-verbose] <name>
```

Returns

New pblock object.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<name>	Name of the new pblock

Categories

XDC, Floorplan

Description

Defines a Pblock to allow you to add logic instances for floorplanning purposes.

You can add logic elements to the Pblock using the `add_cells_to_pblock` command, and then place the Pblocks onto the fabric of the FPGA using the `resize_pblocks` command. The `resize_pblock` command can also be used to manually move and resize Pblocks.

You can nest one Pblock inside another for hierarchical floorplanning using the `set_property` command to define the PARENT property as shown in the second example.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Send Feedback

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - (Required) The name of the Pblock to be created.

Examples

The following example creates a Pblock called pb_cpuEngine, and then adds only the leaf-cells found in the cpuEngine module, clearing placement constraints for placed instances:

```
create_pblock pb_cpuEngine
add_cells_to_pblock pb_cpuEngine [get_cells cpuEngine/*] \
    -add_primitives -clear_locs
```

The following example shows nesting Pblocks, by creating Pblocks called pb_usbEngine0 and pb_usbEngine1, and a third Pblock called pb_usbTop. Then pb_usbEngine0 and pb_usbEngine1 are nested inside pb_usbTop using the `set_property` command:

```
create_pblock pb_usbEngine0
create_pblock pb_usbEngine1
create_pblock pb_usbTop
set_property PARENT pb_usbTop [get_pblocks {pb_usbEngine?}]
```

See Also

- add_cells_to_pblock
- get_pblocks
- resize_pblock
- set_property

create_peripheral

Create a peripheral with a VLNV.

Syntax

```
create_peripheral [-dir <arg>] [-quiet] [-verbose] <vendor> <library>
    <name> <version>
```

Usage

Name	Description
[-dir]	Directory path for remote Peripheral to be created and managed outside the project
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<vendor>	Vendor, for example xilinx.com
<library>	Library, for example ip
<name>	Name, for example myip
<version>	Version, for example 1.4

Categories

Project, IPFlow, CreatePeripheral

Description

Create an AXI peripheral to add to the IP repository with the specified VLNV attribute.

The AXI peripheral that is created is just a framework until interfaces have been added to the peripheral using the `add_peripheral_interface` command, and the peripheral has been generated using the `generate_peripheral` command.

Arguments

`-dir <arg>` - (Optional) Specify an output directory to store the AXI peripheral data files. By default, the peripheral is created and added into the source directory structure, `. ./project_name.srcs/sources_1/ip`, of the current project.

Note: If the AXI peripheral is stored outside of the current project, the specified directory should be added to the IP_REPO_PATH property of the current fileset using the `set_property` command to make the peripheral available through the IP catalog:

```
set_property IP_REPO_PATHS  {C:/Data/axi_peripheral/ourIP_2.1}
[current_fileset]
update_ip_catalog -rebuild
```

Send Feedback

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<vendor> - Defines the vendor portion of the VLNV attribute that defines the location of the AXI peripheral in the IP catalog. The VLNV is the <Vendor:Library:Name:Version> string which identifies the IP in the catalog.

<library> - The library portion of the VLNV attribute.

`<name>` - The name portion of the VLNV attribute.

<version> - The version portion of the VLNV attribute.

Example

The following example creates a new AXI peripheral, with the VLNV attribute as specified:

```
create_peripheral {myCompany.com} {user} {testAXI1} {1.3}
   -dir {C:/Data/new_periph}
```

This example creates a new AXI peripheral, with the VLNV attribute as specified, and captures the peripheral object in a Tcl variable for later processing, then adds AXI slave interfaces to the peripheral:

```
set perifObj [ create_peripheral {myCompany.com} {user} {testAXI1} \
   {1.3} -dir {C:/Data/new_periph} ]
add_peripheral_interface {S0_AXI} -interface_mode {slave} \
   -axi_type {lite} $perifObj
add_peripheral_interface {S1_AXI} -interface_mode {slave} \
   -axi_type {lite} $perifObj
add_peripheral_interface {S2_AXI} -interface_mode {slave} \
   -axi_type {lite} $perifObj
```

See Also

- add_peripheral_interface
- generate_peripheral
- write_peripheral

Send Feedback

create_pin

Create pins in the current design

Syntax

```
create_pin [-from <arg>] [-to <arg>] -direction <arg> [-quiet] [-verbose]
    <pins>...
```

Usage

Name	Description
[-from]	Starting bus index
[-to]	Ending bus index
-direction	Pin direction Values: IN, OUT, INOUT
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<pins>	Names of pins to create

Categories

[Netlist](#)

Description

Add single pins or bus pins to the current netlist of an open Synthesized or Implemented Design. You may define attributes of the pin such as direction and bus width, as well as the pin name.

Bus pins can be created with increasing or decreasing bus indexes, using negative and positive index values.

The pins must be created on an existing cell instance, or it is considered a top-level pin which should be created using the `create_port` command. If the instance name of a cell is not specified as part of the pin name, an error will be returned.

Note: You cannot add pins to library macros, or macro-primitives.

Netlist editing changes the in-memory view of the netlist in the current design. It does not change the files in the source fileset, or change the persistent design on the disk. Changes made to the netlist may be saved to a design checkpoint using the `write_checkpoint` command, or may be exported to a netlist file such as Verilog, VHDL, or EDIF, using the appropriate `write_*` command.

Note: Netlist editing is not allowed on the elaborated RTL design.

Send Feedback

Arguments

`-from <arg>` - (Optional) The starting index of a bus pin.

`-to <arg>` - (Optional) The ending index of a bus pin.

`-direction` - (Required) The direction of the pin. Valid values are IN, OUT, and INOUT.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<pins> - (Required) The name of the pins to create. You must specify the pin names hierarchically from the cell instance the pin is assigned to. Pins created at the top-level of the design are ports, and should be created with the `create_port` command.

Examples

The following example creates a new input pin on the cpuEngine module with the specified pin name:

```
create_pin -direction IN cpuEngine/inPin
```

The following example sets the hierarchy separator, creates a new black box instance of the reference cell, and creates a twenty-four bit bidirectional bus for that instance:

```
set_hierarchy_separator |
create_cell -reference dmaBlock -black_box usbEngine0|myDMA
create_pin -direction INOUT -from 0 -to 23 usbEngine0|myDMA|dataBus
```

See Also

- create_cell
- create_net
- create_port
- connect_net
- disconnect_net
- remove_cell
- remove_pin
- resize_pin_bus

Send Feedback

- set_hierarchy_separator
- write_checkpoint
- write_edif
- write_verilog
- write_vhdl

create_port

Create scalar or bus port

Syntax

```
create_port -direction <arg> [-from <arg>] [-to <arg>] [-diff_pair]
    [-interface <arg>] [-quiet] [-verbose] <name> [<negative_name>]
```

Returns

List of port objects that were created.

Usage

Name	Description
`-direction`	Direction of port. Valid arguments are IN, OUT and INOUT
`[-from]`	Beginning index of new bus
`[-to]`	Ending index of new bus
`[-diff_pair]`	Create differential pair of ports
`[-interface]`	Assign new port to this interface
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<name>`	Name of the port
`[<negative_name>]`	Optional negative name of a diff-pair

Categories

PinPlanning

Description

Creates a port and specifies such parameters as direction, width, single-ended or differential, and optionally assigns it to an existing interface. New ports are added at the top-level of the design hierarchy.

Bus ports can be created with increasing or decreasing bus indexes, using negative and positive index values.

The `create_port` command can be used to create a new port in an I/O Planning project, or while editing the netlist of an open Synthesized or Implemented design.

Netlist editing changes the in-memory view of the netlist in the current design. It does not change the files in the source fileset, or change the persistent design on the disk. Changes made to the netlist may be saved to a design checkpoint using the `write_checkpoint` command, or may be exported to a netlist file such as Verilog, VHDL, or EDIF, using the appropriate `write_*` command.

Note: Netlist editing is not allowed on the elaborated RTL design.

Arguments

`-direction` - (Required) The direction of the port. Valid arguments are IN, OUT, and INOUT.

`-from <arg>` - (Optional) The beginning index of a new bus. A bus can start from a negative index value.

`-to <arg>` - (Optional) The ending index of a new bus. A bus can end on a negative index value.

`-diff_pair` - (Optional) Create the specified port as a differential pair of ports. In this case both a positive and negative side port will be created. If only <name> is specified, the positive side port will be assigned the specified <name>, and the negative side port will be assigned <name_N>. If both <name> and <negative_name> are specified, the positive side port will be assigned <name>, and the negative side port will be assigned <negative_name>.

`-interface <arg>` - (Optional) Assign the port to the specified interface.

Note: The interface must first be defined with the `create_interface` command.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<name> - (Required) The name of the port to create. If `-diff_pair` is specified, <name> is assigned to the positive side port, and the negative side port is <name>_N.

<negative_name> - (Optional) Use this option to specify the name of the negative side port when `-diff_pair` is specified. In this case, <name> will be assigned to the positive side port, and <negative_name> will be assigned to the negative side port.

Examples

The following example creates a new input port, named PORT0:

```
create_port -direction IN PORT0
```

The following example creates a new interface called Group1, and then creates a four-bit, differential pair output bus using the specified interface. Since the bus ports are defined as differential pairs, and only `<name>` is specified, the negative side ports are automatically named D_BUS_N:

```
create_interface Group1
create_port -direction OUT -from 0 -to 3 -diff_pair -interface Group1 D_BUS
```

Note: This command results in the creation of eight ports: D_BUS[0] D_BUS_N[0] D_BUS[1] D_BUS_N[1] D_BUS[2] D_BUS_N[2] D_BUS[3] D_BUS_N[3]

With only `<name>` specified, the following example creates differential pair output ports named data and data_N.

```
create_port -direction OUT -diff_pair data
```

With both `<name>` and `<negative_name>` specified, the following example creates differential pair output ports named data_P and data_N.

```
create_port -direction OUT -diff_pair data_P data_N
```

See Also

- create_interface
- make_diff_pair_ports
- place_ports
- remove_port
- resize_port_bus
- split_diff_pair_ports

create_port_on_reconfigurable_module

Generate a port on a given reconfigurable cell

Syntax

```
create_port_on_reconfigurable_module [-cell <arg>] [-port <arg>]
    [-direction <arg>] [-from <arg>] [-to <arg>] [-quiet] [-verbose]
```

Usage

Name	Description
[-cell]	(Required) specify the HD.RECONFIGURABLE cell name for port punching
[-port]	(Required) specify the newly added port name on given HD.RECONFIGURABLE cell
[-direction]	(Required) specify the direction of ports, it could be either INPUT, OUTPUT or INOUT
[-from]	(Optional) specify the lower boundary of port bus Default: -1
[-to]	(Optional) specify the higher boundary of port bus Default: -1
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

FileIO

create_power_rail

Create a new Power Rail

Syntax

```
create_power_rail [-power_sources <args>] [-direct] [-quiet] [-verbose]
    <name>
```

Returns

New power_rail object.

Usage

Name	Description
[-power_sources]	List of power sources. Can be power rails and/or power supplies
[-direct]	Implies if the power is a direct rail
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<name>	Name of the new power rail

Categories

XDC, Floorplan

Send Feedback

create_pr_configuration

Create new Configuration

Syntax

```
create_pr_configuration -name <arg> [-partitions <args>]
    [-greyboxes <args>] [-use_netlist] [-quiet] [-verbose]
```

Usage

Name	Description
-name	Name of the Configuration
[-partitions]	List of partition instances and reconfig modules pairs
[-greyboxes]	List of instances to which buffers need to be inserted for all ports
[-use_netlist]	Use netlist for getting instances of partition_defs to creating configurations
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Object](#), [Partition](#)

Description

⭐ **IMPORTANT!** *You must first define the project as a Partial Reconfiguration (PR) project by setting the PR_FLOW property on the project to TRUE, or by using the **Tools → Enable Partial Reconfiguration** command.*

The Partial Reconfiguration flow lets you create Partition Definitions (partitionDefs) from hierarchical cells in a design, and to specify reconfigurable modules (RMs) to be assigned to these partitionDefs to create unique configurations of the design based on the combination of the core design and one or more RMs. The PR design flow requires the implementation of each PR configuration, resulting in partial bitstreams for the RMs, but complete bitstreams for each integrated configuration. Refer to the *Vivado Design Suite User Guide: Dynamic Function eXchange* (UG909) for more information.

The `create_pr_configuration` command defines the combination of the static logic and the RM to create a unique configuration of the design. The PR configuration is the design that is implemented and the bitstream is generated for.

You will also need to create implementation runs for the PR configuration using the `create_run -pr_config` command.

This command returns the name of the newly created PR configuration, or returns an error if the command fails.

Arguments

`-name <arg>` - (Required) The name to give to the new PR configuration.

`-partitions <arg>` - (Optional) Specify the partition instance and reconfigurable module to apply to that instance in the PR configuration. The argument must be specified as <partitionInstance>:<RM>, and should be specified as a list when multiple name/value pairs are defined. This format lets you assign different RMs to multiple instances of a single partitionDef in the design.

`-greyboxes <arg>` - (Optional) Indicates that the specified partition instances will be defined as greyboxes, which are populated with LUT1s and signal buffers. Refer to the *Vivado Design Suite User Guide: Dynamic Function eXchange* (UG909) for more information.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Example

The following example defines a new PR configuration and assigns the partitions:

```
create_pr_configuration -name prConfig1 -partitions fftEngine:fftTop
```

The following example defines three PR configurations, with two partitionDefs and different RMs assigned to each partition, as well an empty configuration with greyboxes:

```
create_pr_configuration -name cfg1 -partitions \
[list fftEngine:fftTop mgtEngine:mgtTop]
create_pr_configuration -name cfg2 -partitions \
[list fftEngine:fftTop mgtEngine:mgtBottom]
create_pr_configuration -name cfg3 -partitions { } \
-greyboxes [list fftEngine mgtEngine]
```

See Also

- create_partition_def
- create_reconfig_module
- current_pr_configuration

Send Feedback

- delete_pr_configurations
- get_pr_configurations
- report_pr_configuration_analysis
- setup_pr_configurations

create_project

Create a new project

Syntax

```
create_project [-part <arg>] [-force] [-in_memory] [-ip] [-rtl_kernel]
     [-quiet] [-verbose] [<name>] [<dir>]
```

Returns

New project object.

Usage

Name	Description
[-part]	Target part
[-force]	Overwrite existing project directory
[-in_memory]	Create an in-memory project
[-ip]	Default GUI behavior is for a managed IP project
[-rtl_kernel]	Default GUI behavior is for a RTL Kernel project
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<name>]	Project name
[<dir>]	Directory where the project file is saved Default: .

Categories

[Project](#)

Description

Creates a Vivado Design Suite project file (.xpr), or a project file for the Vivado Lab Edition (.lpr), in the specified directory.

For the Vivado Lab Edition: The `create_project` command has a different command syntax, with fewer options, in the Vivado Lab Edition. The options that are not supported in Vivado Lab Edition are:

- `-part` - The Vivado Lab Edition project (`.lpr`) does not specify a target part because the `current_hw_target` and `current_hw_device` determine the target part.

- `-ip` - The Vivado Lab Edition does not define projects for the Managed IP flow.

Send Feedback

For the Vivado Design Suite: The default project created for the Vivado Design Suite is an RTL project, which defines the project as holding and manage RTL source files in the source fileset. The type of project is determined by the `DESIGN_MODE` Property on the source fileset when the project is created. To change the project type, use the `set_property` command to set the `DESIGN_MODE` property on the `current_fileset` as follows:

- RTL Project - `set_property DESIGN_MODE RTL [current_fileset]`

- Netlist Project - `set_property DESIGN_MODE GateLvl [current_fileset]`

- I/O Planning Project - `set_property DESIGN_MODE PinPlanning [current_fileset]`

Refer to the *Vivado Design Suite User Guide: System-Level Design Entry* (UG895) for more information on the different types of projects.

This command returns a transcript of its process and the name of the created project, or returns an error if it fails.

Arguments

`-part <arg>` - (Optional) Specifies the Xilinx part to be used for the project. This can be changed after the project is created. If the `-part` option is not specified, the default part will be used. This option is not supported in Vivado Lab Edition.

`-force` - (Optional) This option is required to overwrite an existing project. If the project name is already define in the specified <dir> then you must also specify the `-force` option for the tool to overwrite the existing project.

Note: If the existing project is currently open in the tool, the new project will overwrite the existing project on the disk, but both projects will be opened in the tool. In this case you should probably run the `close_project` command prior to running `create_project`.

`-in_memory` - (Optional) Specifies that the project should be created in the in-memory database of the Vivado Design Suite to support the Non-Project design flow. This project will not result in a project file or directory structure being written to disk. The purpose of the in-memory project is to allow properties usually associated with a project-based design to be associated with the in-memory design of the non-project design flow.

> **TIP:** *The use of `-in_memory` is not part of the standard non-project design flow, and may cause unexpected behaviors in the Vivado tools. For more information on Non-Project Mode refer to the Vivado Design Suite User Guide: Design Flows Overview (UG892).*

`-ip` - (Optional) Create a project for the Managed IP flow for exploring IP in the IP catalog, customizing IP, and managing a repository of configured IP. Refer to the *Vivado Design Suite User Guide: Designing with IP* (UG896) for more information on the Managed IP flow. This option is not supported in Vivado Lab Edition.

`-rtl_kernel` - (Optional) Specifies the creation of the project is for use in defining an RTL kernel for the Vitis core development kit.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - (Required) This argument does not require a parameter name, however, it must appear before the specified <dir>. Since these commands do not have parameters, the tool interprets the first argument as `<name>` and uses the second argument as <dir>. A project file is created `<name>.xpr`, (or `<name>.lpr`) and a project data folder is also created `<name>.data` and both are written into the specified directory <dir>.

Note: The project file created by the tool is an RTL source file by default. You must use the `set_property` command to set the DESIGN_MODE property to change the project from an RTL source project to a Netlist or an I/O Pin Planning project.

<dir> - (Optional) This argument specifies the directory name to write the new project file into. If the specified directory does not exist a new directory will be created. If the directory is specified with the complete path, the tool uses the specified path name. However, if <dir> is specified without a path, the tool looks for or creates the directory in the current working directory, or the directory from which the tool was launched.

Note: When creating a project in GUI-mode, the tool appends the `<name>` to the <dir> and creates a project directory <dir>/<name> and places the new project file and project data folder into that project directory.

Examples

When run from the Vivado Design Suite, the following example creates a project called `project1.xpr` in a directory called `myDesigns`:

```
create_project project1 myDesigns
```

Note: Because the <dir> is specified as the folder name only, the tool will create the project in the current working directory, or the directory from which the tool was launched.

When run from the Vivado Lab Edition, this example creates a project called `project1.lpr` in a directory called `myDesigns`:

```
create_project project1 myDesigns
```

Send Feedback

The following example creates a project called Proj1 in a directory called FPGA in C:/Designs. In addition, the tool will overwrite an existing project if one is found to exist in the specified location. In the second and third lines, the location of `-force` is changed to show the flexibility of argument placement.

```
create_project Proj1 C:/Designs/FPGA -force
-or-
create_project Proj1 -force C:/Designs/FPGA
-or-
create_project -force Proj1 C:/Designs/FPGA
```

Note: In all cases the first argument without a preceding keyword is interpreted as the `<name>` variable, and the second argument without a preceding keyword is the <dir> variable.

The following example creates a project for the Manage IP flow in the specified directory:

```
create_project -ip manageIP C:/Data
```

The following example creates a new project called `pin_project`, and then sets the `DESIGN_MODE` property as required for an I/O Pin Planning project, and finally opens an IO design:

```
create_project pin_project C:/Designs/PinPlanning
set_property design_mode PinPlanning [current_fileset]
open_io_design -name io_1
```

See Also

- close_project
- current_hw_device
- current_hw_target
- current_project
- open_io_design
- open_project
- save_project_as
- set_property

create_property

Create property for class of objects(s)

Syntax

```
create_property [-description <arg>] [-type <arg>] [-enum_values <args>]
    [-default_value <arg>] [-file_types <args>] [-display_text <arg>]
    [-quiet] [-verbose] <name> <class>
```

Returns

The property that was created if success, "" if failure.

Usage

Name	Description
[-description]	Description of property
[-type]	Type of property to create; valid values are: string, int, long, double, bool, enum, file Default: string
[-enum_values]	Enumeration values
[-default_value]	Default value of type string
[-file_types]	File type extensions (without the dot)
[-display_text]	Text to display when selecting the file in file browser
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<name>	Name of property to create
<class>	Object type to create property for; valid values are: design, net, cell, pin, port, pblock, interface, fileset

Categories

[PropertyAndParameter](), [XDC]()

Description

Creates a new property of the `<type>` specified with the user-defined `<name>` for the specified `<class>` of objects. The property that is created can be assigned to an object of the specified class with the `set_property` command, but is not automatically associated with all objects of that class.

The `report_property -all` command will not report the newly created property for an object of the specified class until the property has been assigned to that object.

Arguments

`-description <arg>` - (Optional) Provide a description of the property being created. The description will be returned by the Vivado Design Suite help system when the property is queried.

`-type <arg>` - (Optional) The type of property to create. There allowed property types include:

- `string` - Allows the new property to be defined with string values. This is the default value when `-type` is not specified.

- `int` - Allows the new property to be defined with short four-byte signed integer values with a range of -2,147,483,648 to 2,147,483,647. If a floating point value is specified for an `int` property type, the Vivado tool will return an error.

- `long` - Specifies signed 64-bit integers with value range of -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807. If a floating point value is specified for an `long` property type, the tool will return an error.

- `double` - Allows the new property value to be defined with a floating point number.

- `bool` - Allows the new property to be defined as a boolean with a true (1, or yes) or false (0, or no) value.

- `enum` - An enumerated data type, with the valid enumerated values specified by the `-enum_values` option.

- `string_list` - A Tcl list of string values.

- `int_list` - A Tcl list of integer values.

- `double_list` - A Tcl list of floating point values.

`-enum_values <args>` - (Optional) A list of enumerated values that the property can have. The list must be enclosed in braces, {}, with values separated by spaces. This option can only be used with `-type enum`.

`-default_value <args>` - (Optional) The default value to assign to the property. This option can be used for `string`, `int`, `bool`, and `enum` type properties.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - (Required) The name of the property to be defined. The name is case sensitive.

Send Feedback

<class> - (Required) The class of object to assign the new property to. All objects of the specified class will be assigned the newly defined property. Valid classes are: design, net, cell, pin, port, Pblock, interface, and fileset.

Examples

The following example defines a property called PURPOSE for cell objects:

```
create_property PURPOSE cell
```

Note: Because the `-type` was not specified, the value will default to strings.

The following example creates a pin property called COUNT which holds an Integer value:

```
create_property -type int COUNT pin
```

See Also

- get_property
- list_property
- list_property_value
- report_property
- reset_property
- set_property

create_reconfig_module

Create new reconfig Module

Syntax

```
create_reconfig_module -name <arg> [-top <arg>] [-gate_level]
    -partition_def <arg> [-define_from <arg>] [-define_from_file <arg>]
    [-quiet] [-verbose]
```

Usage

Name	Description
-name	Name of the Reconfig Module
[-top]	module name of the top module
[-gate_level]	Create Reconfig Module whcih alllows adding DCP/EDIF files only
-partition_def	PartitionDef in which reconfig module will be created
[-define_from]	Name of the module in the source fileset to be the top of the blockset
[-define_from_file]	full path of the top source file in the source fileset for which reconfigurable module to be created.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Object, Partition

Description

⭐ **IMPORTANT!** *You must first define the project as a Partial Reconfiguration (PR) project by setting the PR_FLOW property on the project to TRUE, or by using the **Tools → Enable Partial Reconfiguration** command.*

The `create_reconfig_module` command defines an reconfigurable module (RM) from a specified hierarchical cell, or design file, and assigns it to the specified Partition Definition (partitionDef) in the current project.

Send Feedback

The Partial Reconfiguration flow allows RMs to be swapped into and out of a partitionDef to create a unique configuration of the design based on the combination of the core design and an RM. A single partitionDef can have multiple RMs to contain different netlists, constraints, or implementations. Each instance of the partitionDef in the design can be assigned a different RM to support many different configurations. The PR design flow requires the implementation of each PR configuration, resulting in partial bitstreams for the RMs, but complete bitstreams for each integrated configuration. Refer to the *Vivado Design Suite User Guide: Dynamic Function eXchange* (UG909) for more information.

This command returns the hierarchical name of the newly created RM, or returns an error if the command failed.

Arguments

`-name <arg>` - (Required) Specify a name for the reconfigurable module (RM) being created.

`-top <arg>` - (Optional) Specify the top-level cell that defines the hierarchy of the RM. The cell can be specified by name.

`-gate_level` - (Optional) Indicates that the RM is defined by a netlist file (EDIF or DCP) rather than by RTL source files.

`-partition_def <arg>` - (Required) Specify the partitionDef object that the RM is assigned to. The partitionDef can be specified by name, or as an object as returned by the `get_partition_defs` command.

`-define_from <arg>` - (Optional) Specify the hierarchical cell in the current project that the RM is defined from. The source files of the specified cell will define the source file contents of the RM. The cell can be specified by name.

`-define_from_file <arg>` - (Optional) Specify the netlist or DCP file that defines the source for gate-level RM. If the path is not specified as part of the filename, the file will be looked for in the current working directory, or the directory that the Vivado tool was launched from.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The example below creates a reconfigurable module with the specified name:

```
create_reconfig_module -name fftBottom -partition_def \
[get_partition_defs partDef1 ] -top fftTop
```

See Also

- create_partition_def
- create_pr_configuration
- delete_reconfig_modules
- get_partition_defs
- get_reconfig_modules
- report_pr_configuration_analysis
- set_property

create_report_config

Create Configurable Report objects.

Syntax

```
create_report_config [-report_name <arg>] [-report_type <arg>]
    -steps <args> -runs <args> [-options <arg>] [-copy_of <args>] [-quiet]
    [-verbose]
```

Returns

List of configurable report objects.

Usage

Name	Description
[-report_name]	Name of configurable report object created. Can not be used when creating multiple objects
[-report_type]	Type of configurable report object(s) created. Not required with '-copy_of'
-steps	List of run step(s) for object(s) created
-runs	List of run(s) for object(s) created
[-options]	options for report command to be set at creation of configurable report object, except with '-copy_of'
[-copy_of]	configurable report object to be copied
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Object](#), [Report](#)

Description

This command lets you create configurable report objects to add to synthesis and implementation runs, or to add to Report Strategies. A report object defines the report type and options that are run every time a specified step of a synthesis or implementation design run is completed. A Report Strategy lets you define a collection of report objects to associate with many synthesis and implementation runs, and reuse at different stages of the design flow. See the *Vivado Design Suite User Guide: Implementation* (UG904) for more information.

Send Feedback

Each report object has the 'OPTIONS.MORE_OPTIONS' property, which lets you specify command line options of the Tcl `report_*` command associated with the report object. These command line options are used when the report is generated during the synthesis or implementation run. You can specify the command line options using the `-options` argument as described below, or by manually setting the 'OPTIONS.MORE_OPTIONS' property of an existing report object using the `set_property` command. Refer to the specific `report_*` command for information on the available command line options.

Arguments

`-report_name <arg>` - (Optional) Specifies the name to assign to the report when it is generated. When the name is not specified, the name will be automatically defined as a combination of the `-runs`, `-steps`, and `-report_type` options.

> 💡 **TIP:** *The `-report_name` option cannot be specified when creating multiple report objects. In this case the report name will be automatically defined.*

`-report_type <arg>` - (Optional) Specifies the report command to be run by the report object. Most of the `report_*` Tcl commands can be specified as the report type to create.

> 💡 **TIP:** *The `-report_type` cannot be specified when using the `-copy_of` option to create a copy of an existing report object.*

`-steps <args>` - (Required) Specifies the synthesis or implementation process steps to associate the report object with. The object can be specified for use with multiple steps to have the report rerun at each step, in which case the name of the report will be automatically defined. Accepted values include all of the available process steps for synthesis or implementation: synth_design, opt_design, place_design, route_design...

`-runs <args>` - (Required) Specifies the synthesis or implementation design run to associate the report object with. The object can be specified for use with multiple design runs, in which case the name of the report will be automatically defined.

`-options <arg>` - (Optional) Specifies various command line options for the specific `report_*` command being run. See the specific report command for the available options. This option can not be specified when using the `-copy_of` option to create a copy of an existing report object.

> ⭐ **IMPORTANT!** *There is no checking provided of the `-options` to ensure the options are correct and applicable to the report type specified. If you indicate options that are incorrect the report will return an error when it is run.*

`-copy_of <arg>` - (Optional) Specifies a report object to use as a template for the new report object. The new report object can be associated with new `-steps` and `-runs`.

> 💡 **TIP:** *The* `-report_type` *and* `-options` *cannot be specified when using the* `-copy_of` *option.*

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example defines a new report object with the specified name and type and associates it with the route_design step of the impl_1 run:

```
create_report_config -report_name post_route_datasheet -report_type
report_datasheet \
-steps route_design -runs impl_1
```

This example is the same as the prior example, except it does not specify the name, and so the report object is named automatically as shown in the return value:

```
create_report_config -report_type report_datasheet -steps route_design -
runs impl_1
impl_1_route_report_datasheet_0
```

The following example creates a new report object for the timing summary report, with the specified command-line options, and associates the object with multiple steps of an implementation run:

```
create_report_config -report_type report_timing_summary \
-steps {opt_design place_design route_design} -runs {impl_2}\
-options {-no_detailed_paths -report_unconstrained}
```

See Also

- delete_report_configs
- generate_reports
- get_report_configs
- set_property

create_run

Define a synthesis or implementation run for the current project

Syntax

```
create_run [-constrset <arg>] [-parent_run <arg>] [-part <arg>] -flow <arg>
    [-strategy <arg>] [-report_strategy <arg>] [-pr_config <arg>] [-quiet]
    [-verbose] <name>
```

Returns

Run object

Usage

Name	Description
[-constrset]	Constraint fileset to use
[-parent_run]	Synthesis run to link to new implementation run
[-part]	Target part
-flow	Flow name
[-strategy]	Strategy to apply to the run
[-report_strategy]	Report strategy to apply to the run
[-pr_config]	partition configuration to apply to the run
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<name>	Name for new run

Categories

Project

Description

Defines a synthesis or implementation run. The attributes of the run can be configured with the use of the `set_property` command.

Arguments

`-constrset <arg>` - (Optional) The constraint set to use for the synthesis or implementation run.

Send Feedback

`-parent_run <arg>` - The run that defines the netlist for the current run. For netlist-based projects the `-parent_run` argument is not required. For an RTL sources project, the parent_run must be specified for implementation runs, but is not required for synthesis runs. For the Partial Reconfiguration flow the `-parent_run` can describe the synthesis run, an implementation run, or the PR configuration as specified by the `-pr_config` option.

`-part <partName>` - (Optional) The Xilinx part to be used for the run. If the `-part` option is not specified, the default part defined for the project will be assigned as the part to use.

`-flow <arg>` - (Required) The tool flow and release version for the synthesis tool, for example {Vivado Synthesis 2017} or the implementation tool {Vivado Implementation 2017}.

`-strategy <arg>` - (Optional) The strategy to employ for the synthesis or implementation run. There are many different strategies to choose from within the tool, including custom strategies you can define. Refer to the appropriate user guide for a discussion of the available synthesis and implementation strategies. If the strategy argument is not specified, "Synthesis Defaults" or "Implementation Defaults" will be used as appropriate.

`-report_strategy <arg>` - (Optional) Specifies a Report Strategy that defines a collection of report objects to run at different stages of the design flow. Report objects are created with the `create_report_config` command. Report strategies are defined in the Settings dialog box of the Vivado IDE.

`-pr_config <arg>` - (Optional) Specifies a partition configuration to apply to the new run. Partition configurations are defined by the `create_pr_configuration` command. Refer to the *Vivado Design Suite User Guide: Dynamic Function eXchange* (UG909) for more information.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - (Required) The name of the synthesis or implementation run to be created.

Examples

The following example creates a run named synth_1 referencing the Vivado synthesis tool flow:

```
create_run -flow {Vivado Synthesis 2013} synth_1
```

Note: The defaults of sources_1, constrs_1, and the default part for the project will be used in the synthesis run. In addition, since this is a synthesis run, the `-parent_run` argument is not required.

Send Feedback

The following example creates an implementation run based on the Vivado Implementation 2013 tool flow, and attaches it to the synth_1 synthesis run previously created:

```
create_run impl_2 -parent_run synth_1 -flow {Vivado Implementation 2013}
```

Note: The `-parent_run` argument is required in this example because it is an implementation of synthesized RTL sources.

See Also

- create_pr_configuration
- create_report_config
- current_run
- launch_runs
- set_property

create_single_pass_run

Exports an intelligent design run result into a single run for the current project

Syntax

```
create_single_pass_run -reference_run <arg> [-quiet] [-verbose] <name>
```

Returns

Run object

Usage

Name	Description
-reference_run	Intelligent design run for which the result has to be exported into a single run
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<name>	Name for new run

Categories

Project

Description

Generates a single pass design run that will be equivalent to the multi-pass Intelligent Design Run. As Intelligent Design Runs can take the equivalent compile time of many single pass runs, the output run of this command allows faster iterations when the design changes.

Arguments

`-reference_run <arg>` - (Required) The top level completed Intelligent Design Run name.

`<name>` - (Required) The new design run name for the single pass run to be created. The run name must not be a run that already exists.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns `TCL_OK` regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Send Feedback

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example creates an implementation run call `new_impl_1` from the completed run
Intelligent Design `Run i_impl_1_1`:

```
create_single_pass_run -reference_run [get_runs i_impl_1_1] new_impl_1
```

See Also

- create_run

Send Feedback

create_slack_histogram

Create histogram

Syntax

```
create_slack_histogram [-to <args>] [-delay_type <arg>] [-num_bins <arg>]
    [-max_paths <arg>] [-slack_less_than <arg>] [-slack_greater_than <arg>]
    [-group <args>] [-significant_digits <arg>] [-scale <arg>]
    [-file <arg>] [-append] [-details] [-no_header] [-name <arg>]
    [-cells <args>] [-quiet] [-verbose]
```

Usage

Name	Description
[-to]	To clock
[-delay_type]	Type of path delay: Values: max, min, min_max Default: max
[-num_bins]	Maximum number of bins: Valid Range (1-100) Default: 10
[-max_paths]	Number of paths to consider for -timing option Default: INT_MAX
[-slack_less_than]	Display paths with slack less than this Default: 1e+30
[-slack_greater_than]	Display paths with slack greater than this Default: -1e+30
[-group]	Limit report to paths in this group(s)
[-significant_digits]	Number of digits to display: Range: 0 to 3 Default: 3
[-scale]	Type of scale on which to draw the histogram; Values: linear, logarithmic Default: linear
[-file]	Filename to output results to. (send output to console if -file is not used)
[-append]	Append the results to file, don't overwrite the results file
[-details]	Report will generate detailed table
[-no_header]	do not generate a report header
[-name]	Output the results to GUI panel with this name
[-cells]	run create_slack_histogram on the specified cell(s)
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Report](), [Timing]()

Description

Create a slack histogram grouping paths into slack ranges, and displaying the results graphically or in text format.

Send Feedback

Inside the text report, this command generates some slack tables based on the timing corners and number of paths that are analyzed. An optional detailed table provides specific information for all the analyzed endpoints.

In GUI report, this command creates a slack histogram grouping paths into slack ranges, and displaying the results graphically.

TIP: *This command provides a graphical slack histogram that requires the tool to be running in GUI mode.*

Arguments

`-to` <args> - (Optional) Specify a clock name, to analyze paths that end in the specified clock domain.

`-delay_type <arg>` - (Optional) Specifies the type of path delay to analyze when creating the slack report. The valid values are min, max, and min_max. The default setting for `-delay_type` is max.

`-num_bins` <args> - (Optional) Specify the number of slack bins to divide the results into. The number of bins determines the granularity of the histogram returned. The range of slack values calculated is divided evenly into the specified number of bins, and the paths are grouped into the bins according to their slack values. The value can be specified as a number between 1 and 100, with a default value of 10.

`-max_paths <arg>` - (Optional) The maximum number of paths to analyze. This is specified as a value greater than or equal to 1. By default the create_slack_histogram command will report all existing critical paths.

`-slack_less_than <arg>` - (Optional) Report slack on paths with a calculated slack value less than the specified value. Can be used with `-slack_greater_than` to provide a range of slack values of specific interest.

`-slack_greater_than <arg>` - (Optional) Report slack on paths with a calculated slack value greater than the specified value. Can be used with `-slack_less_than` to provide a range of slack values of specific interest.

`-group` <args> - (Optional) Report slack for paths in the specified path groups. Currently defined path groups can be determined with the `get_path_groups` command.

`-significant_digits <arg>` - (Optional) The number of significant digits in the output results. The valid range is 0 to 3. The default setting is 3 significant digits.

`-scale [ linear | logarithmic ]` - (Optional) Specify the Y-axis scale to use when presenting the slack histogram. Logarithmic allows for a smoother presentation of greatly different values, but linear is the default.

Note: `-scale` is specific to the GUI report.

`-details` - (Optional) Provide a detailed table of the timing paths that are analyzed. The table lists each endpoint pin or port with the source/destination clock, and timing corner used for analysis.

`-no_header` - (Optional) Do not write a header to the report.

`-file <arg>` - (Optional) Write the report into the specified file. The specified file will be overwritten if one already exists, unless `-append` is also specified.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-name <arg>` - (Optional) Specifies the name of the results set for the GUI. If the name specified is currently opened, the `create_slack_histogram` will overwrite the current results.

`-cells <arg>` - (Option) Generate the report for the specified hierarchical cells. The details of the report will be based on the specified cells rather than the whole design or `current_instance`.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

Create a slack histogram of the current design, using the default values and outputting the results to the named result set in the GUI.

```
create_slack_histogram -name slack1
```

Generate a slack histogram text report (including detailed table) of the current design, using the default values and outputting the results to the specified file.

```
create_slack_histogram -details -file text_report.txt
```

See Also

- delete_timing_results
- get_path_groups
- report_timing

create_sysgen

Create DSP source for Xilinx System Generator and add to the source fileset

Syntax

```
create_sysgen [-quiet] [-verbose] <name>
```

Returns

Name for the new sub module.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<name>	Sub module name

Categories

[SysGen](#)

Description

Create a DSP sub-module for use in the current project, and add it to the source files.

This command will launch System Generator for DSP to let you design the hardware portion of your system design. System Generator is a DSP design tool from Xilinx that allows the RTL source files, Simulink® and MATLAB® software models, and C/C++ components of a DSP system to come together in a single simulation and implementation environment.

For more information on using specific features of the tool refer to *System Generator for DSP Getting Started Guide* (UG639).

You can also add existing DSP model files (.mdl) from System Generator into the current project using the `add_files` command.

The command returns the name of the DSP module created and added to the project.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - (Required) The name of the DSP module to create and add to the current project.

Examples

The following example launches System Generator and allows you to define and configure the specified DSP module:

```
create_sysgen DSP_mod1
```

See Also

- add_files
- generate_target
- list_targets
- make_wrapper

create_waiver

Create a DRC/METHODOLOGY/CDC/LINT message waiver

Syntax

```
create_waiver [-type <arg>] [-id <arg>] [-objects <args>] [-from <args>]
    [-to <args>] [-strings <args>] [-rtl_name <args>]
    [-rtl_hierarchy <args>] [-rtl_file <args>] [-of_objects <args>]
    [-user <arg>] -description <arg> [-tags <arg>] [-timestamp <arg>]
    [-scoped] [-quiet] [-verbose]
```

Returns

Waiver

Usage

Name	Description
[-type]	Type of waiver - DRC, METHODOLOGY, CDC, LINT
[-id]	ID of the DRC/METHODOLOGY/CDC/LINT message being waived, not needed for -of_objects use
[-objects]	List of inserted message objects for which a DRC/METHODOLOGY waiver will be set (i.e. %ELG, %SIG, etc. for cells or nets, etc., sites, etc., or '*CELL', '*NET', '*SITE', etc. as wildcards
[-from]	List of source (driver) pins or ports (or '*PORT', '*PIN' as wildcard) for which a CDC waiver will be set
[-to]	List of target (load) pins or ports (or '*PORT', '*PIN' as wildcard) for which a CDC waiver will be set
[-strings]	List of inserted message string values for which a DRC/METHODOLOGY waiver will be set (i.e. %STR for strings, or '*' as wildcard)
[-rtl_name]	List of rtl_name values for the LINT waiver ('*' as wildcard)
[-rtl_hierarchy]	List of rtl_hierarchy values for the LINT waiver ('*' as wildcard)
[-rtl_file]	List of rtl_file values for the LINT waiver ('*' as wildcard)
[-of_objects]	List of DRC/METHODOLOGY/CDC violation objects for which waiver(s) will be set
[-user]	Name of the user creating the waiver (required, but if not specified, the waivers will take the USER name from the environment if it is available)
-description	Description string of the cause for the waiver
[-tags]	Optional list of one or more tags to aid in subsequent waiver identification or categorization
[-timestamp]	Timestamp of waiver - for restaining the original time of a waiver being (re)created after being written
[-scoped]	Flag waiver to interpret object wildcards as scoped to the current_instance that is set
[-quiet]	Ignore command errors

Send Feedback

Name	Description
`[-verbose]`	Suspend message limits during command execution

Categories

Waiver, XDC, DRC, Methodology, Lint, Object

Description

After `report_drc`, `report_methodology`, or `report_cdc` commands are run, they return messages of specific violations or conditions found in the design. These violations can prevent the design from progressing until they have been resolved or addressed in some way. The `create_waiver` command lets you select individual violations or specific checks that can be waived for a design, letting you move forward in the design flow.

> ⭐ **IMPORTANT!** *Use caution when waiving violations. Waivers may let you proceed in the design flow, but also let you create a design that is fundamentally flawed.*

The user creating the waiver is required to provide a user ID and description in the `create_waiver` command in order to provide some history of the waiver.

A waiver must be specified for an individual DRC or methodology violation, or for a specific DRC or methodology check, or for a CDC path. The waiver must be assigned to a specific object, or specific violation ID, or for paths using `-from/-to` arguments. The form of the `create_waiver` command varies depending on the check, violation, or object being waived, as shown in the examples below.

> 💡 **TIP:** *Although many of the arguments are described as optional, some form of identifier is required to associate the waiver with its target.*

To save waivers from one design session to the next, you must use `write_waivers` to create an XDC file of the waiver commands, and `read_xdc` to read those waivers back into the design when it is reopened.

After creating a waiver, you will need to rerun the DRC, methodology, or CDC report to have the waiver considered in the analysis. To see what waivers are in place in the current design you can use the `report_waivers` command. In addition, the `report_drc`, `report_methodology`, and `report_cdc` commands have options to run the reports on waived violations or checks. Use the `delete_waivers` command to remove waivers from the design.

Arguments

`-type <arg>` - (Optional) Specifies the type of waiver to create. Currently supports DRC, METHODOLOGY, and CDC.

`-id <arg>` - (Optional) Specifies the ID of the check or violation being waived. This is not required when using the `-of_objects` option. The waiver associated with the ID can also be further limited by specifying the `-objects`, `-from/-to`, or the `-strings` arguments.

`-objects <arg>` - (Optional) For DRC and methodology checks and violations, this option specifies the object or list of objects that the waiver applies to. Objects are specified using the convention of the violation definition (e.g. %ELG, %SIG, etc. for cells or nets, etc., sites, etc., or '*CELL', '*NET', '*SITE', etc. as wildcards). Refer to the `create_drc_check` or `create_drc_violation` commands for more information.

`-from <arg>` - (Optional) For CDC checks or violations, this option lists source (driver) pins or ports (or '*PORT', '*PIN' as wildcard) that the waiver applies to.

`-to <arg>` - (Optional) For CDC checks or violations, this option lists target (load) pins or ports (or '*PORT', '*PIN' as wildcard) that the waiver applies to.

`-strings` - (Optional) For DRC and methodology checks and violations, this option specifies inserted message string values (i.e. %STR for strings, or '*' as wildcard) that the waiver applies to.

`-of_objects <arg>` - (Optional) Specifies a list of DRC violations as returned by `get_drc_violations`; or a list of methodology violations as returned by `get_methodology_violations`; or a list of CDC violations as returned by `get_cdc_violations`. The defined waiver applies to the specified list of violation objects.

`-user <arg>` - (Optional) Specifies the user ID of the person responsible for creating the waiver. When not specified, the user ID is obtained from the OS.

`-description <arg>` - (Required) Provides a brief description of the waiver, and why it is being applied.

`-tags <arg>` - (Optional) Specifies a list of user-defined strings to be associated with the waiver. Tags can be used to filter waivers when using the `get_waivers` command.

`-timestamp <arg>` - (Optional) Specifies the original time of the waiver as it is created. Timestamp must be specified in the form "SUN Aug 6 17:02:21 GMT 2017". If `-timestamp` is not specified, the current time is used.

> 💡 **TIP:** *The following Tcl command specifes the timestamp in local time:* `clock format [clock seconds] -gmt 0`

`-scoped` - (Optional) Specify that waivers should be interpreted with object wildcards as scoped to the `current_instance` that is set.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Send Feedback

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

This example creates a waiver for a methodology check based on the specified ID:

```
create_waiver -id TIMING-18 -user samwise -description {waive rule check}
```

The following example creates a DRC check waiver for the indicated ID on the specified list of port objects, and provides the timestamp in local time:

```
create_waiver -type DRC -id UCIO-1 -user samwise -desc {waiving DRC
violation} \
-objects [get_ports {src_in* dest_out*}] \
-timestamp [clock format [clock seconds] -gmt 0]
```

The following example creates a waiver for specific CDC paths in the design, defined by the -from/-to arguments:

```
create_waiver -type CDC -id CDC-6 -user samwise \
-description {Paths to be re-tested later}\
-from [list [get_pins {inst_xpm_grey/src_gray_ff_reg[3]/C \
inst_xpm_grey/src_gray_ff_reg[16]/C \
inst_xpm_grey/src_gray_ff_reg[22]/C \
inst_xpm_grey/src_gray_ff_reg[25]/C}]  ] \
-to [list [get_pins {inst_xpm_grey/dest_graysync_ff_reg[0][1]/D \
inst_xpm_grey/dest_graysync_ff_reg[0][6]/D \
inst_xpm_grey/dest_graysync_ff_reg[0][9]/D \
inst_xpm_grey/dest_graysync_ff_reg[0][24]/D}]  ]
```

See Also

- current_instance
- delete_waivers
- get_cdc_violations
- get_drc_violations
- get_methodology_violations
- get_waivers
- report_cdc
- report_drc
- report_methodology
- report_waivers
- write_waivers

Send Feedback

create_wave_config

Creates a new wave config

Syntax

```
create_wave_config [-quiet] [-verbose] [<name>]
```

Returns

The new wave config.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<name>]	Creates a new wave configuration of the specified name, or a default name if no name given. A new wave window showing that WCFG is also created and made the current wave window

Categories

[Waveform](#)

Description

Create a new wave configuration object in the current simulation, and open the waveform configuration in the Vivado IDE. This will make the new wave configuration object the current wave configuration.

In the Vivado® simulator GUI, you can work with a waveform to analyze your design and debug your code. The Wave Config file contains the list of wave objects (signals, dividers, groups, virtual buses) to display, and their display properties, plus markers. A wave configuration displays with top-level HDL objects, and can be further populated using commands like `add_wave` and `add_wave_divider`. Any changes made to a wave configuration can be saved to a Wave Config file with the `save_wave_config` command.

This command returns the name of the waveform configuration created, or an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - (Optional) The name to give to the wave configuration object. If no `<name>` is specified, an untitled waveform configuration will be created, called "Untitled #", where # represents a numerical sequence beginning at 1.

Examples

The following example creates a new wave configuration object with the specified name:

```
create_wave_config testbench1
```

See Also

- close_wave_config
- current_wave_config
- get_wave_configs
- open_wave_database
- open_wave_config
- save_wave_config

create_xps

Create embedded source for XPS and add to the source fileset (Not supported anymore. Please use Vivado IP integrator.)

Syntax

```
create_xps [-quiet] [-verbose] <name>
```

Returns

Source file name that was created.

Usage

Name	Description
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<name>`	Source name

Categories

Project

Description

Create an Embedded Processor source for use in the current project, and add it to the source files.

This command will launch the Xilinx Platform Studio (XPS) to let you design the hardware portion of the embedded processor system. In XPS you can define and configure the microprocessor, peripherals, and the interconnection of these components. After you exit XPS, the created files for the Embedded Processor sub-design will be written to the local project directory (`<project_name>.srcs/sources_1/edk/<name>`), and added to the source files.

For more information on using specific features of XPS refer to *EDK Concepts, Tools, and Techniques* (UG683).

You can also add existing Xilinx Microprocessor Project (.xmp) files from XPS in the current project using the `add_files` command.

The command returns the name of the Embedded Processor sub-design created.

Send Feedback

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - (Required) The name of the Embedded Processor sub-design to create and add to the current project.

Examples

The following example launches XPS to define and configure the specified Embedded Processor sub-design:

```
create_xps xpsTest1
```

See Also

- add_files
- generate_target
- list_targets
- make_wrapper

current_bd_design

Set or get current design.

Syntax

```
current_bd_design [-quiet] [-verbose] [<design>]
```

Returns

The current design object, "" if failed.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<design>]	Name of current design to be set

Categories

IPIntegrator

Description

Defines the current IP subsystem design for use with the IP integrator of the Vivado Design Suite, or returns the name of the current design in the active project.

The current IP subsystem design and current IP subsystem instance are the target of most of the IP integrator Tcl commands and design changes made in the tool. The current IP subsystem instance can be defined using the `current_bd_instance` command.

You can use the `get_bd_designs` command to get a list of open IP subsystem designs in the active project.

A complete list of IP integrator Tcl commands can be returned using the following command from the Vivado Design Suite Tcl shell:

```
load_features IPIntegrator
help -category IPintegrator
```

Note: The `load_features` command is only needed if the IP integrator is not currently loaded in the Vivado Design Suite.

Send Feedback

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<design> - (Optional) The name of an IP subsystem design to set as the current design in the IP integrator. If a <design> is not specified, the command returns the current IP subsystem design of the active project.

Examples

The following example sets the IP subsystem design as the current design:

```
current_bd_design design_1
```

See Also

- current_bd_instance
- get_bd_designs
- load_features
- open_bd_design

Send Feedback

current_bd_instance

Set or get current cell instance.

Syntax

```
current_bd_instance [-quiet] [-verbose] [<instance>]
```

Returns

The current cell instance object, "" if failed.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<instance>]	Name of current cell instance to be set

Categories

[IPIntegrator](#)

Description

Set or get the current hierarchical cell instance in the current IP integrator subsystem design, as defined by `current_bd_design`. The current instance is referenced from the top-level of the subsystem design hierarchy, or "/".

This command returns the current IP integrator cell instance object, or returns nothing if the command failed.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<instance> - (Optional) The name of the IP integrator hierarchical cell to be set as the current instance of the subsystem design.

- The <instance> is specified relative to the presently defined current instance, using the defined hierarchy separator.

- Use '.' to move up one level of the hierarchy relative to the current instance.

- If the <instance> argument is omitted, the current instance is reset to the top module in the subsystem design hierarchy.

- If the <instance> is specified as '.' then the name of the current instance is returned, and the instance is not changed.

Examples

The following example sets the current instance in the subsystem design to the specified module:

```
current_bd_instance module2
```

The following example returns the current instance:

```
current_bd_instance .
```

This example resets the current instance of the subsystem design to the top level of the hierarchy:

```
current_instance /
```

See Also

- current_bd_design

current_board

Get the current board object

Syntax

```
current_board [-quiet] [-verbose]
```

Returns

Current board object.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Object](), [Board]()

Description

Returns the board in use in the current project.

The board file, `board.xml` located in the `data/boards` folder of the Vivado Design Suite installation area, stores information regarding board attributes. The board provides a representation of the overall system that the Xilinx device is a part of, and can help define key aspects of the design, such as clock constraints, I/O port assignments, and supported interfaces. You can create custom boards by defining a custom Board Interface file, as described in the *Vivado Design Suite User Guide: System-Level Design Entry (UG895)*.

The board can be specified:

- When the project is created by selecting Boards from the Default Part dialog box.

- By setting the BOARD_PART property on the current project as shown in the example.

- By selecting the Project Device in the Settings dialog box in an open project in the Vivado IDE.

Refer to the *Vivado Design Suite User Guide: System-Level Design Entry (UG895)* for information on creating projects, and on configuring project settings.

> **IMPORTANT!** *When you specify the board with the `set_property` command, the target part is also changed to match the part required by the specified BOARD property.*

The `current_board` command returns the `Vendor:Board_Name:File_Version` attributes of the current board, as defined in the BOARD_PART property. The command returns nothing when the project targets a Xilinx FPGA instead of a TRD and board, or when the BOARD_PART property has not been defined. The command returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Example

The following example sets the BOARD property for the current project, then reports the board in use by the project:

```
set_property BOARD_PART xilinx.com:kc705:1.0 [current_project]
current_board
   xilinx.com:kintex7:kc705:1.0
```

This example shows the results of setting the BOARD_PART property, causing the target part to be changed as a result. The target part, as defined in the PART property, is changed automatically, and a warning is returned:

```
set_property BOARD_PART xilinx.com:ac701:1.0 [current_project]
   WARNING: [Project 1-153] The current project part 'xc7k325tffg900-2' does
   not match with the 'XILINX.COM:AC701:1.0' board part settings. The
project
   part will be reset to 'XILINX.COM:AC701:1.0' board part.
```

Note: You can use the `report_property` command to check the BOARD_PART and PART property on the `current_project` to see the changes.

See Also

- current_board_part
- current_project
- get_board_components
- get_board_interface_ports
- get_board_part_interfaces
- get_board_parts

Send Feedback

- get_board_part_pins
- get_boards
- report_property
- set_property

current_board_part

Get the current board_part object

Syntax

```
current_board_part [-quiet] [-verbose]
```

Returns

Current board_part object.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Object, Project, Board

Description

Return the Xilinx device used in the current project or design.

The board file, `board.xml` located in the `data/boards` folder of the Vivado Design Suite installation area, stores information regarding board attributes. The board provides a representation of the overall system that the Xilinx device is a part of, and can help define key aspects of the design, such as clock constraints, I/O port assignments, and supported interfaces. You can create custom boards by defining a custom Board Interface file, as described in the *Vivado Design Suite User Guide: System-Level Design Entry (UG895)*.

The board part provides a representation of the Xilinx device in the context of the board-level system, and is represented by the `part0` component in the Board Interface file.

The board can be specified:

- When the project is created by selecting Boards from the Default Part dialog box.

- By setting the BOARD_PART property on the current project as shown in the example.

- By selecting the Project Device in the Settings dialog box in an open project in the Vivado IDE.

Refer to the *Vivado Design Suite User Guide: System-Level Design Entry (UG895)* for information on creating projects, and on configuring project settings.

Send Feedback

> ⭐ **IMPORTANT!** *When you specify the board with the `set_property` command, the target part is also changed to match the part required by the specified BOARD_PART property.*

The `current_board_part` command returns the NAME property of the current board part. The command returns a warning when the project targets a Xilinx FPGA instead of a board, or when the BOARD_PART property has not been defined. The command returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Example

The following example sets the BOARD_PART property for the current project, then reports the board part in use by the project:

```
set_property BOARD_PART xilinx.com:kc705:part0:1.0 [current_project]
current_board_part
   xilinx.com:kc705:part0:1.0
```

This example shows the results of setting the BOARD_PART property, causing the target part to be changed as well. The target part is changed automatically, and a warning is returned:

```
set_property BOARD_PART xilinx.com:ac701:part0:1.0 [current_project]
WARNING: [Project 1-153] The current project part 'xc7k325tffg900-2'
   does not    match with the 'XILINX.COM:AC701:PART0:1.0' board part
   settings. The project part will be reset to 'XILINX.COM:AC701:PART0:1.0'
   board part.
INFO: [Project 1-152] Project part set to artix7 (xc7a200tfbg676-2)
```

Note: You can use the `report_property` command to check the BOARD_PART and PART property on the `current_project` to see the changes.

Send Feedback

The following example shows how to get DEVICE, PACKAGE, SPEED and FAMILY properties for the part defined by the PART_NAME property of the current board part:

```
get_property DEVICE [get_parts [get_property PART_NAME \
    [current_board_part]]]
get_property PACKAGE [get_parts [get_property PART_NAME \
    [current_board_part]]]
get_property SPEED [get_parts [get_property PART_NAME \
    [current_board_part]]]
get_property FAMILY [get_parts [get_property PART_NAME \
    [current_board_part]]]
```

See Also

- current_project
- get_board_components
- get_boards
- get_board_part_interfaces
- get_board_part_pins
- get_board_parts
- report_property
- set_property

current_design

Set or get the current design.

Syntax

```
current_design [-quiet] [-verbose] [<design>]
```

Returns

Design object

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<design>]	Name of current design to be set

Categories

SDC, XDC

Description

Defines the current design or returns the name of the current design in the active project.

The current design and current instance are the target of most Tcl commands, design edits and constraint changes made in the tool. The current instance can be defined using the `current_instance` command.

You can use the `get_designs` command to get a list of open designs in the active project, and use the `get_projects` command to get a list of open projects.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Send Feedback

Note: Message limits can be defined with the `set_msg_config` command.

<design> - (Optional) The name of design to set as the current design. If a <design> is not specified, the command returns the current design of the active project.

Examples

The following example sets the design rtl_1 as the current design:

```
current_design rtl_1
```

See Also

- current_instance
- get_designs
- get_projects

Send Feedback

current_fileset

Get the current fileset (any type) or set the current fileset (applicable to simulation filesets only)

Syntax

```
current_fileset [-constrset] [-simset] [-quiet] [-verbose] [<fileset>...]
```

Returns

Current fileset (the current srcset by default).

Usage

Name	Description
[-constrset]	Get the current constraints fileset
[-simset]	Get the current active simulation fileset
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<fileset>]	Specify the simulation fileset to set as current (active); optional

Categories

[Project](#)

Description

Get the active source, constraint, or simulation fileset within the current project.

When used without any options, current_fileset sets and returns the sources_1 set as the active fileset.

This command can also be used to set the current simulation fileset.

Note: Use `set_property CONSTRSET` to define the active constraint set on a synthesis or implementation run.

Arguments

`-constrset` - (Optional) Return the currently active constraint set.

`-simset` - (Optional) Return or set the currently active simulation fileset.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<fileset> - (Optional) The name of the simulation fileset to make active. This argument sets the active simulation fileset in projects with multiple filesets. When <fileset> is not specified, the sources_1 fileset is returned as the active fileset.

Examples

The following example returns the name of the currently active constraint fileset:

```
current_fileset -constrset
```

The following example sets `sim_2` as the active simulation set:

```
current_fileset -simset sim_2
```

See Also

- create_fileset
- delete_fileset
- get_filesets

current_frame

Get index of the selected subprogram frame (default, top i.e. most recent subprogram call) in the call-stack of the HDL process scope (current_scope). Sets current stack frame for the subprogram call-stack of the current_scope.

Syntax

```
current_frame [-up] [-down] [-set <arg>] [-quiet] [-verbose]
```

Returns

Returns index of the selected subprogram frame in the call stack of the current_scope.

Usage

Name	Description
`[-up]`	Selects stack frame of the caller subprogram/process as the current frame.
`[-down]`	Selects stack frame of the callee subprogram as the current frame.
`[-set]`	Selects stack frame with given index as the current frame of the call stack of current HDL process scope. Default: 0
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution

Categories

[Simulation](#)

Description

Returns the index of the frame which is selected as the "current" frame in sub-program call-stack of the current HDL process-scope, or `current_scope`. By default, the most recently called subprogram-frame is the current frame, i.e. with frame-index "0" (zero) and marked with (->).

Switches like `-up`,`-down`, and `-set` let you select other frames in the call-stack, rather than the current frame.

> ⭐ **IMPORTANT!** *The* `current_frame` *strictly follows the* `current_scope`*. If* `current_scope` *is not an HDL process scope waiting inside a sub-program, the* `current_frame` *command reports that the current process does not have an associated sub-program stack.*

Arguments

`-up` - (Optional) Set stack-frame of the CALLER process/subprogram "1" stack frame(hop) away from current subprogram frame, as the current frame.

`-down` - (Optional) Set stack-frame of the CALLEE sub-program "1" stack frame (hop) away from current process/subprogram-frame, as the current frame.

`-set` <num> - (Optional) Set frame with index <num> of the current selected call stack as the current frame. `-set 0` will set the frame stack back to the deepest level of the selected call-stack.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

Example design:

```
module top;
    int i;
    function void f(input int in1);
        automatic int a;
        a = in1 + 7;
        $display($time, " in f :: a %d in1 %d ", a, in1);
    endfunction
    task automatic t(input int in2);
        int b;
        b = in2 + 10;
        $display($time, " in t :: in2 %d b %d ", in2, b);
        #5;
        f(b);      // Case C
        $display($time, " Back in t : after wait and f(%d) ", b);
    endtask
    initial begin                         // "/top/Initial18_0"
        $display($time, " in initial 1 ");
        i = 200;
        t(i);          // Case B
        $display($time, " Back in initial 1 after t(%d) ", i);
    end
    initial begin                         // "/top/Initial25_1"
        $display($time, " in initial 2 ");
        #2;
        f(50);         // Case A
        $display($time, " Back in initial 2 after f(50) ");
    end
endmodule
```

When simulation is stopped inside function "f" for its call at 'Case C', function "f" is called from task "t" at 'Case C' which itself is called from process "/top/Initial18_0" at 'Case B'

```
  > current_scope
      /top/Initial18_0
  > report_frames
    -> 0 : f
        1 : t
        2: /top/Initial18_0
1. > current_frame
     0
2. > current_frame -up
     1
   > report_frames
        0 : f
    -> 1 : t
        2 : /top/Initial18_0
 3. > current_frame -down
     0
    > report_frames
    -> 0 : f
        1 : t
        2: /top/Initial18_0
4. > current_frame -set 1
     1
   > report_frames
        0 : f
    -> 1 : t
        2 : /top/Initial18_0
 5. > current_frame -verbose
     -> 0 : f @top.v:6
        1 : t @top.v:15
        2: /top/Initial18_0 @top.v:21
```

See Also

- current_scope
- report_frames

current_hw_cfgmem

Get or set the current hardware cfgmem

Syntax

```
current_hw_cfgmem [-hw_device <args>] [-quiet] [-verbose] [<hw_cfgmem>]
```

Returns

Hardware cfgmem

Usage

Name	Description
[-hw_device]	list of hardware devices
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<hw_cfgmem>]	list of hardware cfgmems Default: current hardware cfgmem

Categories

Hardware

Description

Set or return the current hardware cfgmem object.

The process whereby the design specific data is loaded or programmed into the Xilinx® FPGA is called configuration. The `create_hw_cfgmem` command defines a flash memory device used for configuring and booting the FPGA device.

When a new hw_cfgmem object is created, it becomes the current hw_cfgmem object. You can use the `current_hw_cfgmem` to return the current hw_cfgmem object, or you can specify a hw_cfgmem object, as returned by `get_hw_cfgmems`, to change the current object.

After the hw_cfgmem object is created, and associated with the hw_device, the configuration memory can be programmed with the bitstream and other data from a memory configuration file created with the `write_cfgmem` command.

The hw_cfgmem object is programmed using the `program_hw_cfgmem` command.

This command returns the current hardware cfgmem as an object, or returns an error if it fails.

Send Feedback

Arguments

`-hw_device <arg>` - (Required) Specify the hw_device object to set the current_hw_cfgmem on. The hw_device must be specified as an object as returned by the `current_hw_device` or `get_hw_devices` commands.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_cfgmem> - (Optional) Specify the hw_cfgmem object to use as the current configuration memory for programming and debug. The hw_cfgmem must be specified as an object as returned by the `get_hw_cfgmems` command.

Example

The following example returns the current hardware cfgmem object:

```
current_hw_cfgmem
```

See Also

- create_hw_cfgmem
- current_hw_device
- get_hw_cfgmems
- get_hw_devices
- program_hw_cfgmem
- write_cfgmem

current_hw_device

Get or set the current hardware device

Syntax

```
current_hw_device [-quiet] [-verbose] [<hw_device>]
```

Returns

Hardware device

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<hw_device>]	hardware device to set as current; optional

Categories

Hardware

Description

Set or return the current Xilinx FPGA targeted by the Hardware Manager in the Vivado Design Suite for programming and debug.

The hardware target is a system board containing a JTAG chain of one or more Xilinx devices that you can program with a bitstream file, or use to debug your design. Connections between hardware targets on the system board and the Vivado Design Suite are managed by the `hw_server` application, and the `connect_hw_server` command. Refer to *Vivado Design Suite User Guide: Programming and Debugging (UG908)* for a list of supported JTAG download cables and devices.

Each hardware target can have one or more Xilinx devices to program, or to use for debugging purposes. The current device is specified or returned by the `current_hw_device` command.

To access a Xilinx FPGA through the Hardware Manager, you must use the following Tcl command sequence:

1. `open_hw` - Opens the Hardware Manager in the Vivado Design Suite.

2. `connect_hw_server` - Makes a connection to a local or remote Xilinx hardware server application.

Send Feedback

3. `current_hw_target` - Defines the hardware target of the connected server.

4. `open_hw_target` - Opens a connection to the hardware target.

5. `current_hw_device` - Specifies the Xilinx FPGA to use for programming and debugging.

After connecting to the appropriate hardware device, you can program the device with a bitstream file using the `program_hw_device` command, and debug the device using any of a number of Hardware Manager Tcl commands. To interactively debug the device open the Hardware Manager in the Vivado Design Suite IDE.

> **IMPORTANT!** *You can use the `current_hw_server`, `current_hw_target`, and `current_hw_device` commands to set the hardware for programming and debugging the design. You should exercise care when using these commands to insure that the current server, target, and device are in sync. The current device should be found on the current target, which should be found on the current server.*

This command returns the current hardware device as an object, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_device> - (Optional) Specify the hw_device object to set as the current device for programming and debug. The hw_device must be specified as an object as returned by the `get_hw_devices` command. If the hardware device is not specified, the `current_hw_device` will be returned.

Example

The following example returns the currently targeted hw_device:

```
current_hw_device
```

This example sets the `current_hw_device` object, then sets the PROGRAM.FILE property for the device, and programs the device:

```
current_hw_device [lindex [get_hw_devices] 0]
set_property PROGRAM.FILE {C:/Data/design.bit} [current_hw_device]
program_hw_devices [current_hw_device]
```

Send Feedback

See Also

- connect_hw_server
- current_hw_target
- get_hw_devices
- get_hw_targets
- open_hw_target

current_hw_ila

Get or set the current hardware ILA

Syntax

```
current_hw_ila [-quiet] [-verbose] [<hw_ila>]
```

Returns

Hardware ILA

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<hw_ila>]	hardware ILA

Categories

Hardware

Description

Set or return the current hardware ILA debug core targeted by the Hardware Manager in the Vivado Design Suite for programming and debug.

The Integrated Logic Analyzer (ILA) debug core lets you perform in-system debug of implemented designs, or design bitstreams, on a programmed Xilinx FPGA. The ILA core includes many advanced features of modern logic analyzers, including boolean trigger equations, and edge transition triggers. You can use the ILA core to probe specific signals of the design, to trigger on programmed hardware events, and capture data from the Xilinx FPGA in real-time. Refer to *LogiCORE IP Integrated Logic Analyzer (PG172)* for details of the ILA core.

ILA debug cores can be added to a design instantiating an ILA core from the IP catalog into the RTL design, or using the `create_debug_core` command to add the ILA core to the synthesized netlist. Refer to *Vivado Design Suite User Guide: Programming and Debugging (UG908)* for more information on adding ILA debug cores to the design.

After generating a bitstream from the design, and programming the device with the `program_hw_devices` command, the ILA debug cores in the design are accessible from the Hardware Manager using the `get_hw_ilas` command. The debug probes assigned to the ILA debug cores in the design can be returned with the `get_hw_probes` command.

This command returns the current hardware ILA core as an object, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_ila> - (Optional) Specify the hw_ila object to set as the current debug core for programming and debug. The hw_ila must be specified as an object as returned by the `get_hw_ilas` command. If a ILA debug core is not specified, the `current_hw_ila` will be returned.

Example

The following example returns the current hardware ILA debug core:

```
current_hw_ila
```

See Also

- current_hw_device
- get_hw_devices
- get_hw_ilas
- get_hw_ila_datas

current_hw_ila_data

Get or set the current hardware ILA data

Syntax

```
current_hw_ila_data [-quiet] [-verbose] [<hw_ila_data>]
```

Returns

Hardware ILA data.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<hw_ila_data>]	hardware ILA data

Categories

Hardware

Description

Set or return the current ILA debug core data object.

The ILA data object is created in the Vivado logic analyzer using the `upload_hw_ila_data` command, or the `read_hw_ila_data` command. By default, the current hw_ila_data object is the latest one created by the Vivado logic analyzer. The `current_hw_ila_data` command can be used to change that object.

The ILA debug core captures sample data in real-time as the hardware device runs, based on the event triggers or capture conditions defined on the hw_ila object. The hw_ila object triggers on the hw_device are armed by the `run_hw_ila` command.

The ILA data object can be displayed in the waveform window of the Vivado tools logic analyzer using the `display_hw_ila_data` command. You can also write the ILA data to disk with the `write_hw_ila_data` command to save the ILA debug information for later user and analysis.

This command returns the captured hardware ILA debug core data as an object, or returns an error if it fails.

Send Feedback

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_ila_data> - (Optional) Specify the hw_ila_data object to set as the current data object for programming and debug. The hw_ila_data object must be specified as an object as returned by the `get_hw_ila_datas` command. If the hardware ILA data is not specified, the `current_hw_ila_data` object will be returned.

Example

The following example returns the current data object for the hardware ILA debug core:

```
current_hw_ila_data
```

See Also

- current_hw_ila
- display_hw_ila_data
- get_hw_ilas
- get_hw_ila_datas
- read_hw_ila_data
- run_hw_ila
- write_hw_ila_data

Send Feedback

current_hw_server

Get or set the current hardware server

Syntax

```
current_hw_server [-quiet] [-verbose] [<hw_server>]
```

Returns

Hardware server

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<hw_server>]	hardware server

Categories

[Hardware](#)

Description

Defines the current hardware server from the list of hardware servers that are connected to the Vivado Design Suite, or returns the currently connected hardware server object.

Hardware servers are instances of the Xilinx hardware server (`hw_server`) application running remotely, or on the local machine. The hardware server manages connections to a hardware target, such as a hardware board containing a JTAG chain of one or more Xilinx devices to be used for programming and debugging your FPGA design.

Hardware servers are connected to the Vivado Design Suite with the `connect_hw_server` command. The current hardware server, and the current hardware target and device are the focus of most Hardware Manager Tcl commands. The current target and device can be defined using the `current_hw_target` and `current_hw_device` commands.

Note: There is usually a current hw_server defined, either the last connected hardware server, or one you have defined with this command. However, if you disconnect the current hardware server, you will need to define a new current hw_server object.

You can get a list of connected hardware servers using the `get_hw_servers` command. You can get a list of available hardware targets and devices using the `get_hw_targets` and `get_hw_devices` commands respectively.

Send Feedback

This command returns a hw_server object. If the <hw_server> is specified as part of the `current_hw_server` command, the specified server is defined as the current hardware server and that object is returned. If no server is specified, the `current_hw_server` command returns the current hardware server object.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_server> - (Optional) The hw_server to set as the current hardware server. If the server is not specified, the command will return the currently defined hw_server. The hardware server can be specified by name, or specified as a hw_server object returned by the `get_hw_servers` command.

Example

The following example sets the current hardware server to the specified name:

```
current_hw_server picasso
```

The following returns the current hardware server:

```
current_hw_server
```

See Also

- connect_hw_server
- current_hw_device
- current_hw_target
- disconnect_hw_server
- get_hw_devices
- get_hw_servers
- get_hw_targets
- refresh_hw_server

current_hw_target

Get or set the current hardware target

Syntax

```
current_hw_target [-quiet] [-verbose] [<hw_target>]
```

Returns

Hardware target

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<hw_target>]	hardware target

Categories

Hardware

Description

After opening the Hardware Manager in the Vivado Design Suite, and connecting to the Xilinx hardware server (`hw_server`) using the `connect_hw_server` command, you will need to set the hardware target. This command sets or returns the current hardware target.

The hardware target is a system board containing a JTAG chain of one or more Xilinx devices that you can program with a bitstream file, or use to debug your design. Connections between hardware targets on the system board and the Vivado Design Suite are managed by the hw_server object. Refer to *Vivado Design Suite User Guide: Programming and Debugging (UG908)* for a list of supported JTAG download cables and devices.

The available hardware targets are defined when the Vivado tools Hardware Manager is connected to a hw_server. You can return a list of the available hardware targets using the `get_hw_targets` command, and define the current hardware target using the `current_hw_target` command.

If the <hw_target> is specified as part of the `current_hw_target` command, the specified target is defined as the current hardware target and that object is returned. If no hardware target is specified, the `current_hw_target` command returns the current hardware target object.

Send Feedback

Each hardware target can have one or more Xilinx devices to program, or to use for debugging purposes. The current device is specified or returned by the `current_hw_device` command. After specifying the current hardware target, you can open the connection through the hardware target, to the Xilinx device using the `open _hw_target` command.

> ⭐ **IMPORTANT!** *You can use the* `current_hw_server`, `current_hw_target`, *and* `current_hw_device` *commands to set the hardware for programming and debugging the design. You should exercise care when using these commands to insure that the current server, target, and device are in sync. The current device should be found on the current target, which should be found on the current server.*

This command returns the current hardware target as an object, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_target> - (Optional) Specify the hw_target object to set as the current_hw_target for programming and debug. The hw_target must be specified as an object as returned by the `get_hw_targets` command. If the hardware target is not specified, the `current_hw_target` will be returned.

Example

The following example returns the available hardware targets on the connected hardware servers, and sets the `current hw_target` to the specified target:

```
get_hw_targets
    trumpet/xilinx_tcf/Digilent/210203327985A
    picasso/xilinx_tcf/Digilent/210203368518A
current_hw_target [lindex [get_hw_targets] 1]
```

See Also

- close_hw_target
- current_hw_device
- get_hw_devices
- get_hw_targets
- open_hw_target

Send Feedback

- refresh_hw_target

Send Feedback

current_instance

Set or get the current instance

Syntax

```
current_instance [-quiet] [-verbose] [<instance>]
```

Returns

Instance name

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<instance>]	Name of instance

Categories

SDC, XDC

Description

Set the current instance in the design hierarchy to the specified instance cell or to the top of the current design. By default, `current_instance` points to the top module of the `current_design`, which is not an instantiated cell object. You can also set `current_instance` to reference an instantiated hierarchical cell in the design.

> ⭐ **IMPORTANT!** *Since the top module is not an instantiated object,* `current_instance` *returns an empty string rather than a design object, when set to the top-level of the current design.*

The current design and current instance are the target of most of the commands and design changes you will make. The current design can be defined using the `current_design` command.

You must specify the <instance> name relative to the currently defined instance, and use the established hierarchy separator to define instance paths. You can determine the current hierarchy separator with the `get_hierarchy_separator` command.

Use '..' to traverse up the hierarchical instance path when specifying the current instance.

This command returns the name of the design object of the current_instance, or returns nothing when set to the top of current design.

Send Feedback

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<instance> - (Optional) The name of the instance to be set as the current instance of the current design.

- The <instance> is specified relative to the presently defined current instance, using the defined hierarchy separator.
- Use '.' to move up one level of the hierarchy relative to the current instance.
- If the <instance> argument is omitted, the current instance is reset to the top module in the design hierarchy.
- If the <instance> is specified as '.' then the name of the current instance is returned, and the instance is not changed.

Examples

The following example sets the current instance to the top module of the current design:

```
current_instance
    INFO: [Vivado 12-618] Current instance is the top level of design
'netlist_1'.
```

In this example you have selected an object in the Vivado IDE, and you want to set that selected object as the current instance:

```
current_instance [lindex [get_selected_objects] 0]
```

Note: `get_selected_objects` returns a list, even of one object, so you must use `lindex` to specify an object from that list.

The following example first sets the hierarchy separator character, and then sets the current instance relative to the presently defined current instance:

```
set_hierarchy_separator |
current_instance ..|cpu_iwb_dat_o|buffer_fifo
```

Send Feedback

The following example returns the name of the presently defined current instance:

```
current_instance .
cpuEngine|cpu_iwb_dat_o|buffer_fifo
```

See Also

- current_design
- get_hierarchy_separator
- get_selected_objects
- set_hierarchy_separator

Send Feedback

current_pr_configuration

Get a list of PartitionDefs

Syntax

```
current_pr_configuration [-quiet] [-verbose] [<config>...]
```

Returns

List of PartitionDef objects.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<config>]	Specify the PR configuration to be set as current (active); optional

Categories

Object, Partition

Description

Get or set the current PR configuration.

In the Partial Reconfiguration (PR) design flow, the PR configuration lets you specify a reconfigurable module (RM) to assign to a specific instance of a Partition Definition (partitionDef). This flow lets you create unique configurations of the design based on the combination of the core design and one or more RMs. The PR design flow requires the implementation of each PR configuration, resulting in partial bitstreams for the RMs, but complete bitstreams for each integrated configuration. Refer to the *Vivado Design Suite User Guide: Dynamic Function eXchange* (UG909) for more information.

The current_pr_configuration either returns the PR configuration that is the current or active configfuration in the design, or lets you specify a PR configuration to make active.

This command returns the name of the current PR configuration, or returns an error if the command fails.

Send Feedback

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<config> - (Optional) Specify a PR configuration to make current. The configuration can be specified by name or returned as an object by `get_pr_configurations`. If the configuration is not specified, the current PR configuration is returned.

Example

The following example sets the current PR configuration as specified:

```
current_pr_configuration clockHigh
```

See Also

- create_partition_def
- create_pr_configuration
- create_reconfig_module
- delete_pr_configurations
- get_pr_configurations
- setup_pr_configurations

Send Feedback

current_project

Set or get current project

Syntax

```
current_project [-quiet] [-verbose] [<project>]
```

Returns

Current or newly set project object.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<project>]	Project to set as current

Categories

Project

Description

Specifies the current project or returns the current project when no project is specified.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the set_msg_config command.

<project> - (Optional) The name of the project to make current. This command can be used prior to the close_project to make a specific project active and then to close the project.

Send Feedback

Examples

The following example sets `project_2` as the current project:

```
current_project project_2
```

This command makes the current project the focus of all the tool commands. In the GUI mode, the current project is defined automatically when switching the GUI between projects.

The following example returns the name of the current project in the tool:

```
current_project
```

Note: The returned value is the name of the project and not the name or path of the project file.

See Also

- close_project
- create_project
- current_design

current_run

Set or get the current run

Syntax

```
current_run [-synthesis] [-implementation] [-quiet] [-verbose] [<run>]
```

Returns

Run object

Usage

Name	Description
[-synthesis]	Set or get the current synthesis run
[-implementation]	Set or get the current implementation run (default unless '-synthesis' is specified)
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<run>]	Run to set as current; optional

Categories

[Project](#)

Description

Defines the current synthesis or implementation run, or returns the name of the current run. The current run is the one automatically selected when the Synthesize or Implement commands are launched.

You can use the `get_runs` command to determine the list of defined runs in the current design.

Arguments

`-synthesis` - (Optional) Specifies that the `current_run` command should set or return the name of the current synthesis run.

`-implementation` - (Optional) Specifies that the `current_run` command should set or return the name of the current implementation run. This is the default used when neither `-synthesis` or `-implementation` are specified.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Send Feedback

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<run>` - (Optional) Sets the name of the synthesis or implementation run to make the current run.

Examples

The following example defines the synth_1 run as the current_run:

```
current_run synth_1
```

Note: The `-synthesis` and `-implementation` arguments are not required because the name allows the tool to identify the specific run of interest.

The following command returns the name of the current implementation run:

```
current_run -implementation -quiet
```

See Also

- create_run
- get_runs
- launch_runs

current_scope

Get the current scope or set the current scope

Syntax

```
current_scope [-quiet] [-verbose] [<hdl_scope>]
```

Returns

The current scope.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<hdl_scope>]	Default: NULL

Categories

Simulation

Description

Return the current scope in the current simulation, or set the current scope to the specified HDL scope.

The `current_scope` command returns the name of the current simulation scope.

If <hdl_scope> is supplied then, the current scope is set to the specified scope.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Send Feedback

<hdl_scope> - (Optional) Specify an HDL scope to assign as the current scope of the current simulation. The scope can be specified as an absolute hierarchical path name (such as `/tb/UUT`), a relative path name (`/uut`, `gt`, `./gt`, `uut/fg`) of a scope, or an HDL scope object, returned by the `get_scopes` command.

> 💡 **TIP:** *If you specify '/' as the HDL scope, the scope is reset to the top-level scope in the current simulation.*

Examples

The following example sets the current scope to the specified HDL scope:

```
current_scope /testbench/dut
```

This example returns the current scope name to console:

```
current_scope
```

See Also

- get_scopes
- report_scopes

Send Feedback

current_sim

Set the current simulation object or get the current simulation object

Syntax

```
current_sim [-quiet] [-verbose] [<simulationObject>]
```

Returns

Returns the current simulation object.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<simulationObject>]	Simulation Object to set the current simulation object to Default: NULL

Categories

Simulation

Description

Get or set the current Vivado simulation object.

This command can be used after the Vivado simulator has been launched to return or set the current simulation object.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Send Feedback

<simulationObject> - (Optional) The name of the simulation object to set as the current simulation. If a <simulationObject> is not specified, the command returns the current simulation of the current project.

Examples

The following example sets the curent simulation:

```
current_sim simulation_2
```

See Also

- close_sim
- launch_simulation

current_time

Report current simulation time

Syntax

```
current_time [-s] [-quiet] [-verbose]
```

Returns

Prints the current simulation time on the console in textual format.

Usage

Name	Description
[-s]	Short -- suppress the space between numeric part and unit part
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Simulation

Description

Returns the current simulation time to the Tcl Console or Vivado Design Suite Tcl shell.

Arguments

-s - (Optional) Suppress the space between the numeric portion of the time and the unit portion of the time to make it easier to parse.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the set_msg_config command.

Examples

The following example returns the current time of the current simulation:

```
current_time
```

See Also

- restart
- run
- stop

current_vcd

Return the current VCD object or make VCDObject the current VCD object

Syntax

```
current_vcd [-quiet] [-verbose] [<VCDObject>]
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<VCDObject>]	VCDObject Default: NULL

Categories

Simulation

Description

Defines the current Value Change Dump (VCD) object, or returns the name of the current VCD object in the current simulation.

A VCD file must be opened and assigned to a VCD object using the `open_vcd` command in order for there to be a current VCD object.

This command returns the current VCD object.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<VCDObject> - (Optional) The name of the VCD object to set as the current object. If a <VCDObject> is not specified, the command returns the current VCD object of the active simulation.

Send Feedback

Examples

The following example sets the specified VCD object as current:

```
current_vcd vcd2
```

See Also

- [open_vcd](#)

current_vivado_preferences

Returns the vivado prefernces object.

Syntax

```
current_vivado_preferences [-quiet] [-verbose]
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

vivado_preferences

current_wave_config

Gets the current WCFG object and sets it to the specified WCFG object if given

Syntax

```
current_wave_config [-quiet] [-verbose] [<wcfgObj>]
```

Returns

Returns the new or current wave configuration object .

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<wcfgObj>]	Sets the current WCFG object to the given value of wcfgObj. Defaults to current

Categories

Waveform

Description

Set or get the current wave configuration object for the current simulation.

In the Vivado® simulator GUI, you can work with a waveform to analyze your design and debug your code. A wave configuration displays with top-level HDL objects, and can be further populated using commands like `add_wave` and `add_wave_divider`.

This command returns the name of the current wave configuration object.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Send Feedback

<wcfgObj> - (Optional) Specify a wave configuration object to set as the current wave configuration. The wave configuration object can either be specified by name, or as an object returned by the `get_wave_configs` command.

Examples

The following example gets the testbench wave config object and makes it the current wave configuration in the simulation:

```
current_wave_config [get_wave_config testbench]
```

See Also

- create_wave_config
- get_wave_configs
- open_wave_config

decrypt_bitstream

Decrypt an AES-GCM encrypted bitstream

Syntax

```
decrypt_bitstream -encrypted_file <arg> -keyfile <arg> [-force] [-quiet]
    [-verbose] <file>
```

Usage

Name	Description
`-encrypted_file`	Input AES-GCM encrypted bitstream (.bit or .rbt)
`-keyfile`	File containing encryption keys
`[-force]`	Overwrite existing file
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<file>`	Output decrypted bitstream (.bit, .bin or .rbt)

Categories

FileIO

Description

During implementation of secure encrypted UltraScale architecture designs, bitstream-level verification must be performed on the final bitstream against the "golden" bitstream of the Xilinx tested Security Monitor (SecMon) IP.

The `decrypt_bitstream` command takes an AES-GCM encrypted bitstream file (`.bit` or `.rbt`) from an implemented design that incorporates the SecMon IP, and an encryption key file (`.nky` or `.nkz`), and returns an unencrypted bitstream file. The decrypted bitstream can then be used to complete the bitstream verification process.

This command returns the requested file if successful, or returns an error if it fails.

Arguments

`-encrypted_file <arg>` - (Required) Specifies the AES-GCM encrypted bitstream (`.bit` or `.rbt`) to be decrypted.

`-keyfile <arg>` - (Required) Specifies the name of the encryption key file (`.nky` or `.nkz`). This is necessary for decrypting the encrypted bitstream.

`-force` - (Optional) Overwrite an existing file of the same name.

Send Feedback

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file>` - (Required) Write the decrypted bitstream in standard format (`.bit`) or without header information (`.bin`).

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

Example

The following example decrypts the specified encrypted bitstream:

```
decrypt_bitstream -encrypted_file C:/Data/myDesign.bit \
    -keyfile C:/Data/key.nky -force C:/Data/myDesign_decrypted.bit
```

See Also

- write_bitstream

delete_bd_objs

Delete specified objects.

Syntax

```
delete_bd_objs [-quiet] [-verbose] <objects>...
```

Returns

Pass if successful in deleting objects.

Usage

Name	Description
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<objects>`	The objects to be deleted

Categories

[IPIntegrator](#)

Description

Delete specified objects from the current IP integrator subsystem design.

Objects must be passed directly to the `delete_bd_objs` command, and not simply referenced by the object name. Pins are passed to the command by `get_bd_pins`, for instance, rather than by pin name.

This command returns nothing if it is successful, and returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<objects>` - A list of objects to delete from the current IP integrator subsystem design.

Example

The following example deletes the various objects from the current subsystem design:

```
delete_bd_objs [get_bd_nets /Net] [get_bd_nets /vidout_1_vtg_ce] \
[get_bd_nets /newMod1/aclk_1] [get_bd_ports /addr] [get_bd_cells /vidOut_1]
```

The following example deletes the same objects, but uses multiple `delete_bd_objs` commands to clarify the objects that are being deleted by grouping them by type:

```
delete_bd_objs [get_bd_nets /Net] [get_bd_nets /vidout_1_vtg_ce] \
[get_bd_nets /newMod1/aclk_1]
delete_bd_objs [get_bd_ports /addr]
delete_bd_objs [get_bd_cells /vidOut_1]
```

See Also

- get_bd_addr_segs
- get_bd_addr_spaces
- get_bd_cells
- get_bd_intf_ports
- get_bd_intf_pins
- get_bd_nets
- get_bd_pins
- get_bd_ports

Send Feedback

delete_clock_networks_results

Clear a set of clock networks results from memory

Syntax

```
delete_clock_networks_results [-quiet] [-verbose] <name>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<name>	Name for the set of results to clear

Categories

Report

Description

Clear the results of the specified `report_clock_networks` report from the named result set.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - (Required) Specifies the name of the clock networks results to be cleared.

Examples

The following example clears the specified results set from memory:

```
delete_clock_network_results ClkNets
```

Send Feedback

See Also

- report_clock_networks

Send Feedback

delete_dashboard_gadgets

Deletes the gadgets from project summary dashboard

Syntax

```
delete_dashboard_gadgets [-quiet] [-verbose] <gadgets>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<gadgets>	Gadgets to delete

Categories

Project

Description

This command removes the gadget from the Project Summary dashboard, and removes it from the project.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<gadgets> - (Required) Specifies the dashboard gadgets to delete. Gadgets can be specified by name, or returned by the `get_dashboard_gadgets` command.

Example

The following example deletes the specified gadget:

```
delete_dashboard_gadget gadget_power
```

Send Feedback

See Also

- create_dashboard_gadget
- get_dashboard_gadgets
- move_dashboard_gadget

delete_debug_core

Delete a debug core

Syntax

```
delete_debug_core [-quiet] [-verbose] <cores>...
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<cores>	Debug cores to delete

Categories

Debug

Description

Removes Vivado Lab Edition debug cores from the current project that were added by the `create_debug_core` command.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<cores> - (Required) One or more debug core names to remove from the current project.

Examples

The following command deletes the myCore debug core from the current project:

```
delete_debug_core myCore
```

Send Feedback

The following command deletes all debug cores from the current project:

```
delete_debug_core [get_debug_cores]
```

Note: The `get_debug_cores` command returns all debug cores as a default.

See Also

- create_debug_core
- get_debug_cores

delete_debug_port

Delete debug port

Syntax

```
delete_debug_port [-quiet] [-verbose] <ports>...
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<ports>	Debug ports to delete

Categories

Debug

Description

Deletes ports from Vivado Lab Edition debug cores in the current project. You can disconnect a signal from a debug port using `disconnect_debug_port`, or remove the port altogether using this command.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<ports> - (Required) The <core_name>/<port_name> of the debug port to be removed from the core.

Send Feedback

Examples

The following example deletes the DATA port from myCore:

```
delete_debug_port myCore/DATA
```

Note: Some ports cannot be deleted because an ILA port requires one CLK port and one TRIG port as a minimum.

The following example deletes the trigger ports (TRIG) from the myCore debug core:

```
delete_debug_port [get_debug_ports myCore/TRIG*]
```

Note: This example will not delete all TRIG ports from myCore, because an ILA core must have at least one TRIG port. The effect of this command will be to delete the TRIG ports starting at TRIG0 and removing all of them except the last port.

See Also

- disconnect_debug_port
- get_debug_ports

Send Feedback

delete_drc_check

Delete one or more user-defined DRC checks.

Syntax

```
delete_drc_check [-quiet] [-verbose] <name>...
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<name>	Specify the key for the check to remove. This is the typically of the form PREFIX-id where PREFIX is a 4-6 letter abbreviation and id is a unique identifier. Use get_drc_checks to determine the correct name to use. Only user-defined DRC checks may be deleted.

Categories

DRC, Object

Description

Delete a single user-defined design rule check from the current project. User-defined design rule checks are created using the `create_drc_checks` command.

Note: You cannot delete factory defined rule checks.

Once it has been deleted there is no way to recover a rule check. The `undo` command will not work.

Note: This command returns nothing if successful, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - (Required) Specify the name of a user-defined design rule check to be deleted from the current project.

Examples

The following example deletes the specified design rule check:

```
delete_drc_check LJH-1
```

See Also

- create_drc_check

delete_drc_ruledeck

Delete one or more user defined DRC rule deck objects

Syntax

```
delete_drc_ruledeck [-regexp] [-nocase] [-filter <arg>] [-quiet] [-verbose]
    [<patterns>]
```

Returns

Drc_ruledeck

Usage

Name	Description
`[-regexp]`	Patterns are full regular expressions
`[-nocase]`	Perform case-insensitive matching. (valid only when -regexp specified)
`[-filter]`	Filter list with expression
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`[<patterns>]`	Match the 'drc_ruledeck' objects against patterns. Default: *

Categories

[DRC](), [Object]()

Description

Delete one or more user-defined drc_ruledeck objects from the current project. The rule deck does not have to be empty to be deleted, and once it is deleted there is no way to recover it. The `undo` command will not restore a deleted rule deck.

Note: You cannot delete factory defined rule decks.

A rule deck is a collection of design rule checks grouped for convenience, to be run with the `report_drc` command at different stages of the FPGA design flow, such as during I/O planning or placement. The tool comes with a set of factory defined rule decks, but you can also create new user-defined rule decks with the `create_drc_ruledeck` command.

Note: This command returns nothing if successful, or returns an error if it fails.

Send Feedback

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by the search pattern, based on specified property values. You can find the properties on an object with the `report_property` or `list_property` commands.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Delete the drc_ruledeck objects that match the specified patterns. The default pattern is the wildcard '*' which deletes all user-defined rule decks from the current project. More than one pattern can be specified to delete multiple rule decks based on different search criteria.

Note: You must enclose multiple search patterns in braces, {}, or quotes, "", to present the list as a single element.

Examples

The following example deletes all user-defined rule decks from the current project:

```
delete_drc_ruledeck
```

See Also

- create_drc_ruledeck
- list_property
- report_property

Send Feedback

delete_fileset

Delete a fileset

Syntax

```
delete_fileset [-merge <arg>] [-quiet] [-verbose] <fileset>
```

Usage

Name	Description
[-merge]	Fileset to merge files from the deleted fileset into
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<fileset>	Fileset to be deleted

Categories

[Project](#), [Simulation](#)

Description

Deletes the specified fileset. However, if the fileset cannot be deleted, then no message is returned.

Arguments

`-merge <arg>` - (Optional) Specify a different fileset to merge the files from the deleted fileset into. If the `-merge` option is not specified, then all files in the fileset are removed from the project.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<fileset> - (Required) The name of the fileset to delete. The last constraint or simulation fileset will not be deleted, and no error will be returned under these circumstances.

Examples

The following example deletes the sim_2 fileset from the current project.

```
delete_fileset sim_2
```

Note: The fileset and all of its files are removed from the project. The files are not removed from the hard drive.

See Also

- create_fileset
- current_fileset

delete_hw_axi_txn

Delete hardware AXI Transaction objects

Syntax

```
delete_hw_axi_txn [-quiet] [-verbose] <hw_axi_txns>...
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hw_axi_txns>	hardware AXI Transaction object to delete

Categories

Hardware

Description

This command deletes the named AXI transaction objects, hw_axi_txn, from the specified hw_axi objects.

The `create_hw_axi_txn` command cannot create an object of the same name as an existing object. Use this command to delete any existing objects prior to creating new AXI transaction objects.

This command returns nothing if successful, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_axi_txns> - (Required) The hw_axi_txn objects to delete. The hw_axi_txn must be specified as an object returned by the `get_hw_axi_txns` command.

Send Feedback

Example

The following example deletes the hw_axi_txn object associated with the specified hw_axi:

```
delete_hw_axi_txn [get_hw_axi_txns readAxi1]
```

See Also

- create_hw_axi_txn
- get_hw_axis
- get_hw_axi_txns
- refresh_hw_axi
- reset_hw_axi

Send Feedback

delete_hw_bitstream

Removes the HW Bitstream object from a list of hardware devices.

Syntax

```
delete_hw_bitstream [-of_objects <args>] [-quiet] [-verbose]
```

Returns

Hardware devices

Usage

Name	Description
[-of_objects]	Get 'hw_bitstream' objects of these types: 'hw_device'.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Hardware, Object

Description

This command deletes the hw_bitstream object from the specified hw_device objects.

This clears the PROGRAM.HW_BITSTREAM and PROGRAM.FILE properties on the hw_device objects, and deletes the hw_bitstream object.

Arguments

`-of_objects <arg>` - (Optional) Delete the hardware bitstream object (hw_bitstream) associated with the specified hardware devices (hw_devices). The targets must be specified as objects using the `get_hw_devices` or the `current_hw_device` commands.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Send Feedback

Note: Message limits can be defined with the `set_msg_config` command.

Example

The following example deletes the hw_bitstream object associated with the current hardware device:

```
delete_hw_bitstream -of_objects [current_hw_device]
```

See Also

- create_hw_bitstream
- current_hw_device
- get_hw_devices
- program_hw_devices
- set_property
- write_bitstream

delete_hw_cfgmem

Removes hw_cfgmem object from memory

Syntax

```
delete_hw_cfgmem [-quiet] [-verbose] <cfgmem>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<cfgmem>	Valid hw_cfgmem object

Categories

[Hardware](#)

Description

Removes the specified hw_cfgmem object from the current hw_device.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<cfgmem> - (Required) Specify a hw_cfgmem object to remove from the current hw_device. The hw_cfgmem must be specified as an object as returned by `get_hw_cfgmems` or `current_hw_cfgmem`, rather than simply by name.

Example

The following example removes the current hw_cfgmem object:

```
delete_hw_cfgmem [current_hw_cfgmem]
```

Send Feedback

See Also

- create_hw_cfgmem
- current_hw_device
- get_hw_cfgmems
- get_hw_devices
- get_property
- program_hw_cfgmem

delete_hw_probe

Delete hardware probe objects

Syntax

```
delete_hw_probe [-quiet] [-verbose] <hw_probes>...
```

Usage

Name	Description
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<hw_probes>`	hardware probe objects to delete

Categories

Hardware

Description

Delete a user-defined probe from the current hw_ila. The user-define probe must be created by the `create_hw_probe` command.

This command returns nothing if successful, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_probe> s - Specify user-defined hw_probe objects to delete from the current ILA core. Probes must be specified as objects returned by the `get_hw_probes` command.

Send Feedback

Examples

The following example deletes a user-defined hw_probe object on the current ILA core:

```
delete_hw_probe [get_hw_probe probeAR]
```

See Also

- create_hw_probe
- current_hw_ila
- get_hw_ilas
- get_hw_probes

delete_hw_target

Delete a hw_target

Syntax

```
delete_hw_target [-quiet] [-verbose] [<target_object>]
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<target_object>]	hardware target object to delete Default: current_hw_target

Categories

Hardware

Description

This command deletes a virtual hardware target from the current_hw_server.

The hw_target object must be a virtual target created by the `create_hw_target` command, or an error will be returned.

This command returns nothing if successful, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<target_object> - (Optional) Specifies the SVF or virtual target to delete. If the target is not specified the Vivado hardware manager will attempt to delete the `current_hw_target`.

Send Feedback

Examples

The following example deletes the specified hw_target:

```
delete_hw_target [lindex [get_hw_targets] 1]
```

See Also

- create_hw_target

delete_interface

Delete I/O port interfaces from the project

Syntax

```
delete_interface [-all] [-quiet] [-verbose] <interfaces>...
```

Usage

Name	Description
[-all]	Also remove all of the ports and buses belonging to the interface
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<interfaces>	I/O port interfaces to remove

Categories

PinPlanning

Description

Deletes an existing interface and optionally deletes all of the associated ports and buses using the interface.

Arguments

`-all` - (Optional) Delete all ports, buses, or nested interfaces associated with the specified interface.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<interfaces> - (Required) The name of interfaces to delete.

Send Feedback

Examples

The following example deletes the specified interface and all of its associated ports and buses:

```
delete_interface USB0
```

See Also

- create_interface
- create_port

Send Feedback

delete_ip_run

Deletes the block fileset and run associated with a given IP.

Syntax

```
delete_ip_run [-force] [-quiet] [-verbose] <objects>
```

Usage

Name	Description
[-force]	Force the deletion of the block fileset and run.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<objects>	All of the IP objects (from get_ips or get_files) for which the block fileset and run will be deleted.

Categories

[Project](), [IPFlow]()

Description

Deletes the out-of-context (OOC) synthesis and implementation runs for the specified IP module.

The contents of the run directory are deleted from the project as well as the run. However, the output products created by the run and copied to the IP sources folder, the DCP file and Verilog and VHDL structural netlists, are not deleted from the project. You must use the `reset_target` or `generate_target` command to update the IP output products.

> ⭐ **IMPORTANT!** *The command requires an IP object as specified by the `get_ips` or `get_files` command, and will not delete a run based on either the name of the run, or a run object as returned by `get_runs`.*

Arguments

`-force` - Force the deletion of the run.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<objects>` - (Required) The IP object or file to delete the run from. An IP can be specified as an object using the `get_ips` command, or as a file (XCI) using the `get_files` command.

Examples

The following example deletes the OOC synthesis and implementation runs from the specified IP module:

```
delete_ip_run [get_ips add1]
```

Note: In this example, all run results will also be removed from the run directory on the hard drive.

See Also

- create_ip_run
- generate_target
- get_files
- get_ips
- reset_target

delete_macros

Delete a list of macros

Syntax

```
delete_macros [-quiet] [-verbose] <macros>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<macros>	Macros to delete

Categories

[XDC](XDC)

Description

Delete one or more macro defined by the `create_macro` command.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<macros> - (Required) Specify the name or names of the macros to delete.

Examples

The following example deletes a macro called usbMacro1:

```
delete_macros usbMacro1
```

Send Feedback

See Also

- create_macro

delete_partition_defs

Delete existing PartitionDefs

Syntax

```
delete_partition_defs [-merge <arg>] [-quiet] [-verbose] <partition_defs>
```

Usage

Name	Description
[-merge]	Fileset to merge files into from the default RM of deleted Partition Def
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<partition_defs>	List of PartitionDefs to delete

Categories

[Partition](#)

Description

Delete the specified Partition Definition (partitionDef) objects from the current project.

This command returns a transcript of the file merge process, returns nothing without file merge, or returns an error if the command fails.

Arguments

`-merge <arg>` - (Optional) Specify the name of a fileset to merge files from the default Reconfigurable Module (DEFAULT_RM) of a deleted partitionDef object. The files will be moved from the default RM to the specified fileset.

> **TIP:** *If `-merge` is not specified, then all files in the default RM are removed from the project.*

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<partition_defs> - (Required) Specify one or more partitionDef objects to remove from the current project. The partitionDef objects can be specified by name, or as objects returned by the `get_partition_defs` command.

Example

The following example deletes all the partitionDefs from the current design, merging files from the default RMs of each partition into the source fileset for the design:

```
delete_partition_defs -merge sources_1 [get_partition_defs]
```

See Also

- create_partition_def
- get_partition_defs

delete_pblocks

Remove Pblock

Syntax

```
delete_pblocks [-hier] [-quiet] [-verbose] <pblocks>...
```

Usage

Name	Description
[-hier]	Also delete all the children of Pblock
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<pblocks>	Pblocks to delete

Categories

[Floorplan](#), [XDC](#)

Description

Deletes the specified Pblocks from the design. Pblocks are created using the `create_pblock` command.

Arguments

`-hier` - (Optional) Specifies that Pblocks nested inside the specified Pblock should also be deleted. If the parent Pblock is deleted without the `-hier` option specified, the nested Pblocks will simply be moved up one level.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<pblocks> - (Required) One or more Pblocks to be deleted.

Send Feedback

Examples

The following example deletes the specified Pblock as well as any Pblocks nested inside:

```
delete_pblocks -hier cpuEngine
```

See Also

- create_pblock

delete_power_rails

Delete a Power Rail

Syntax

```
delete_power_rails [-quiet] [-verbose] <power_rail>
```

Usage

Name	Description
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<power_rail>`	power rail to delete

Categories

XDC, Floorplan

delete_power_results

Delete power results that were stored in memory under a given name.

Syntax

```
delete_power_results -name <arg> [-quiet] [-verbose]
```

Usage

Name	Description
-name	Name for the set of results to clear
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Power

Description

Deletes the power analysis results for the specified results set.

Note: This command operates silently and does not return direct feedback of its operation

Arguments

`-name <arg>` - (Required) The name of the results set to delete. This name was either explicitly defined, or was automatically defined when the `report_power` command was run.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Send Feedback

Examples

The following example runs power analysis, and then clears the results:

```
report_power -name my_set
delete_power_results -name my_set
```

See Also

- power_opt_design
- report_power
- reset_switching_activity
- set_switching_activity

Send Feedback

delete_pr_configurations

Delete existing configurations

Syntax

```
delete_pr_configurations [-quiet] [-verbose] <configs>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<configs>	List of Configurations to delete

Categories

[Partition](#)

Description

Delete the specified PR configuration from the current project.

This command returns nothing if successful, or returns an error if the command fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<configs> - (Required) Specify one or more PR configuration objects to remove from the current project. The configurations can be specified by name, or as objects returned by the `get_pr_configurations` command.

Send Feedback

Example

The following example deletes the specified PR configuration:

```
delete_pr_configurations [get_pr_configurations clockHigh]
```

See Also

- create_pr_configuration
- get_pr_configurations
- setup_pr_configurations

delete_qor_suggestions

Deletes a list of available QoR suggestions

Syntax

```
delete_qor_suggestions [-quiet] [-verbose] [<IDs>]
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<IDs>]	Match suggestion names against given names

Categories

Object, Feasibility, Timing

Description

Removes QoR suggestion objects that are currently in memory. Suggestion objects are generated by running the `report_qor_suggestion` command without specifying `-of_objects`, and by reading an RQS file using the `read_qor_suggestions` command. You can get QoR objects by using the `get_qor_suggestions` command.

Arguments

<IDs> - (Required) A list of suggestion object IDs. IDs can be returned using the `get_qor_suggestions` command.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following deletes all the suggestion objects in memory:

```
delete_qor_suggestions [get_qor_suggestions]
```

The following deletes only the Utilization suggestions:

```
delete_qor_suggestions [get_qor_suggestions -filter {CATEGORY==Utilization}]
```

See Also

- get_qor_suggestions
- read_qor_suggestions
- report_qor_assessment
- write_qor_suggestions

delete_reconfig_modules

Delete existing reconfig modules

Syntax

```
delete_reconfig_modules [-merge <arg>] [-quiet] [-verbose] <rms>
```

Usage

Name	Description
[-merge]	Fileset to merge files into from the deleted Reconfig Module
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<rms>	List of Reconfig Modules to delete

Categories

Partition

Description

Delete the specified reconfigurable modules (RMs) from the current project.

This command returns nothing if successful, or returns an error if the command fails.

Arguments

`-merge <arg>` - (Optional) Specify the name of a fileset to merge files from the deleted RM. The files will be moved from the deleted RM to the specified fileset.

> **TIP:** *If `-merge` is not specified, then all files in the deleted RM are removed from the project.*

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<rms> - (Required) Specify one or more RM objects to remove from the current project. The RMs can be specified by name, or as objects returned by the `get_reconfigurable_modules` command.

Example

The following example deletes the specified RM:

```
delete_reconfig_modules usbBlock
```

See Also

- create_reconfig_module
- get_reconfig_modules

delete_report_configs

Delete a set of existing configurable report objects

Syntax

```
delete_report_configs [-quiet] [-verbose] <report_configs>...
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<report_configs>	List of configurable report objects to delete

Categories

Object, Report

Description

Removes specified report objects from the current project. The report objects are created by the `create_report_config` command.

This command returns nothing if successful, or an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<report_configs> - (Required) Specifies the list of report objects to remove from the current project. The reports must be specified as objects returned by the `get_report_configs` command.

Send Feedback

Examples

The following example deletes the specified report_config object:

```
delete_report_configs [get_report_configs post_route_datasheet]
```

See Also

- create_report_config
- get_report_configs

Send Feedback

delete_rpm

Delete an RPM

Syntax

```
delete_rpm [-quiet] [-verbose] <rpm>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<rpm>	RPM to delete

Categories

Floorplan

Description

Deletes the specified Relationally Placed Macro (RPM) from the design.

An RPM is a list of logic elements (FFS, LUT, CY4, RAM, etc.) collected into a set (U_SET, H_SET, and HU_SET). The placement of each element within the set, relative to other elements of the set, is controlled by Relative Location Constraints (RLOCs). Logic elements with RLOC constraints and common set names are associated in an RPM. Refer to the Constraints Guide (UG625) for more information on defining these constraints.

Only user-defined RPMs can be deleted from the design. RPMs defined by the hierarchy or defined in the netlist cannot be deleted by this command.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<rpm> - (Required) The RPM to be deleted.

Examples

The following example deletes the specified RPM (cs_ila_0/U0) from the design:

```
delete_rpm cs_ila_0/U0
```

delete_runs

Delete existing runs

Syntax

```
delete_runs [-noclean_dir] [-quiet] [-verbose] <runs>
```

Usage

Name	Description
[-noclean_dir]	Do not remove all output files and directories from disk
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<runs>	Run to modify

Categories

Project

Description

Deletes the specified runs from the project, and deletes all results of the run from the project directory on the hard drive unless otherwise specified.

Arguments

`-noclean_dir` - Do not delete the run results from the hard drive. The run will be deleted from the project, but the run files will remain in the project directory.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<runs> - (Required) The names of the synthesis or implementation runs to delete from the project.

Send Feedback

Examples

The following example deletes the first_pass run from the project:

```
delete_runs first_pass
```

Note: In this example, all run results will also be removed from the project directory on the hard drive.

The following command deletes the first_pass run, but leaves the run results on the hard drive:

```
delete_runs -noclean_dir first_pass
```

See Also

- create_run
- current_run

delete_timing_results

Clear a set of timing results from memory

Syntax

```
delete_timing_results [-type <arg>] [-quiet] [-verbose] <name>
```

Usage

Name	Description
[-type]	Type of timing results to clear; Values: bus_skew, check_timing, clock_interaction, clock_domain_crossings, config_timing, datasheet, pulse_width, slack_histogram, timing_path, timing_summary
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<name>	Name for the set of results to clear

Categories

[Report](), [Timing]()

Description

Clear the specified timing results from the named result set. Both the type of the timing report, and the name of the timing report must be specified, or the command will fail.

Arguments

`-type <arg>` - (Optional) Specifies the type of timing results to be cleared. The available types are:

- bus_skew - Delete the named `report_bus_skew` report .

- check_timing - Delete the named `check_timing` report.

- clock_interaction - Delete the named `report_clock_interaction` report.

- clock_domain_crossing - Delete the named CDC report.

- config_timing - Delete the current Timing COnfig report.

- datasheet - Delete the named report_datasheet report.

- pulse_width - Delete the named `report_pulse_width` report.

- slack_histogram - Delete the named `create_slack_histogram` report.

Send Feedback

- timing_path - Delete the named `report_timing` report.

- timing_summary - Delete the named `report_timing_summary` report.

Note: The default `-type` is timing_path, to delete reports generated by the `report_timing` command.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - (Required) Specifies the name of the timing results to be cleared.

Examples

The following example clears the specified results set from memory:

```
delete_timing_results -type clock_interaction clkNets
```

See Also

- check_timing
- create_slack_histogram
- report_bus_skew
- report_clock_interaction
- report_cdc
- report_config_timing
- report_datasheet
- report_pulse_width
- report_timing
- report_timing_summary

delete_utilization_results

Delete utilization results that were stored in memory under a given name.

Syntax

```
delete_utilization_results -name <arg> [-quiet] [-verbose]
```

Usage

Name	Description
-name	Name for the set of results to clear
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Report](#)

Description

Clear the specified utilization results from the named result set.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the set_msg_config command.

-name <arg> - (Required) Specifies the name of the results to be cleared.

Examples

The following example clears the specified results set from memory:

```
delete_utilization_results -name SSO1
```

Send Feedback

See Also

- report_utilization

Send Feedback

delete_waivers

Delete one or more DRC/METHODOLOGY/CDC message waivers

Syntax

```
delete_waivers [-scoped] [-quiet] [-verbose] [<objects>...]
```

Usage

Name	Description
`[-scoped]`	flag waiver to interpret object wildcards as scoped to the current_instance that is set
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`[<objects>]`	List of waiver objects, or a list of message objects (cells, nets, sites, etc.) for which DRC/METHODOLOGY waiver(s) were set

Categories

Waiver, Object

Description

After `report_drc`, `report_methodology`, or `report_cdc` commands are run, they return messages of specific violations or conditions found in the design. These violations can prevent the design from progressing until they have been resolved or addressed in some way. The `create_waiver` command lets you select individual violations or specific checks that can be waived for a design, letting you move forward in the design flow.

Use the `delete_waivers` command to remove waivers from the design.

Arguments

`-scoped` - (Optional) Specify that waivers should be interpreted with object wildcards as scoped to the `current_instance` that is set.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Send Feedback

Note: Message limits can be defined with the `set_msg_config` command.

`<objects>` - (Optional) List of waiver objects as returned by `get_waivers`.

Examples

This example deletes all the DRC waivers in the design:

```
delete_waivers [get_waivers -type DRC]
```

See Also

- create_waiver
- current_instance
- get_waivers
- report_cdc
- report_drc
- report_methodology

describe

Describe an HDL object (variable, signal, wire, or reg) by printing type and declaration information

Syntax

```
describe [-quiet] [-verbose] <hdl_object>
```

Returns

The description of the selected objects.

Usage

Name	Description
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<hdl_object>`	The hdl_object or hdl_scope to describe

Categories

[Simulation](#)

Description

Describe an HDL object (variable, signal, wire, or reg) by printing type and declaration information, as well as path, and file information for the HDL source of the specified objects.

> 💡 **TIP:** *The* `describe` *command works for a single HDL object. Use the* `report_objects` *command for a brief report on multiple HDL objects.*

HDL objects include HDL signals, variables, or constants as defined in the Verilog or VHDL test bench and source files. An HDL signal includes Verilog wire or reg entities, and VHDL signals. Examples of HDL variables include Verilog real, realtime, time, and event. HDL constants include Verilog parameters and localparams, and VHDL generic and constants.

The command returns the description of specified HDL objects, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Send Feedback

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hdl_object> - (Required) Specifies a single HDL object to describe.

Note: Objects can be specified by name, or returned as objects by the `get_objects` command.

Examples

The following example shows how the objects description depends on the scope of the current simulation:

```
current_scope testbench
/testbench
describe leds_n
    Signal: {leds_n[3:0]}
    Path: {/testbench/leds_n}
    Location: {File "C:/Data/ug937/sim/testbench.v" Line 9}
current_scope dut
/testbench/dut
describe leds_n
    Port(OUT): {LEDS_n[3:0]}
    Path: {/testbench/dut/LEDS_n}
    Location: {File "C:/Data/sources/sinegen_demo.vhd" Line 42}
```

See Also

- current_scope
- get_objects
- report_objects

detect_hw_sio_links

Automatically detect links between RX and TX endpoints. Create a new link group to contain the links.

Syntax

```
detect_hw_sio_links [-force] [-quiet] [-verbose]
```

Returns

A new hardware SIO link group of found links.

Usage

Name	Description
[-force]	Remove all existing links before detecting links
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Hardware

Description

Automatically detects existing or previously defined communication pathways between GT transmitters and receivers that are defined on the open hardware target.

You can use this command if you change board connections while the serial I/O analyzer is running. The detection algorithm uses changing transmit patterns and detects links on received patterns to determine how GTs are connected to one another on the open hardware target.

A transmitter or receiver of an individual GT on the IBERT debug core can only be used in one hw_sio_link at a time, so the command will not check GTs that are used in existing links. The `-force` option lets you clear all existing links before scanning the open hardware target to check all GTs.

The `detect_hw_sio_links` command defines the found links, and creates a link group to associate the new links.

This command returns the number of links found and the created hw_sio_linkgroup object, or returns an error if it fails.

Arguments

`-force` - (Optional) Delete existing link objects and re-detect links by examining all GTs to ensure the current hardware configuration is properly reflected in the link definitions.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Example

The following example examines the IBERT debug core on the open hardware target to look for existing communication links:

```
detect_hw_sio_links
```

Note: Without the `-force` option, GTs used in existing links will not be examined.

See Also

- create_hw_sio_link
- create_hw_sio_linkgroup
- current_hw_device
- get_hw_sio_iberts
- get_hw_sio_links
- get_hw_sio_linkgroups
- remove_hw_sio_link
- remove_hw_sio_linkgroup

Send Feedback

disconnect_bd_intf_net

Disconnect an intf_net.

Syntax

```
disconnect_bd_intf_net [-quiet] [-verbose] <intf_net> <objects>...
```

Returns

TCL_OK, TCL_ERROR if failed.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<intf_net>	The IntfNet that the objects connect to
<objects>	The objects to disconnect from the intf_net

Categories

IPIntegrator

Description

Disconnect a single interface net in the IP integrator subsystem design from the specified objects. An interface is a grouping of signals that share a common function in the IP integrator subsystem design.

This command lets you disconnect the specified interface net from pins or ports in the IP subsystem design, without deleting the whole net. To delete the whole net, you should use the `delete_bd_objs` command.

This command returns TCL_OK if it is successful, or TCL_ERROR if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

Send Feedback

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<intf_net> - (Required) Specifies a single interface net in the IP subsystem design to disconnect from the specified objects. The net can be specified by name, or as a single object returned by the `get_bd_intf_nets` command.

<objects> - (Required) The list of interface pin or port objects to disconnect the net from. The interface pins and ports must be specified as design objects returned by the `get_bd_intf_pins` or `get_bd_intf_ports` commands. They cannot simply be referenced by name.

Example

The following example disconnects the only interface net in the IP subsystem design, from all interface pin and port objects that are connected to it:

```
disconnect_bd_intf_net [get_bd_intf_nets] \
[get_bd_intf_pins -of_objects [get_bd_intf_nets]] \
[get_bd_intf_ports -of_objects [get_bd_intf_nets]]
```

Note: In this example you know there is only one interface net in the design, or an error would be returned by the `disconnect_bd_intf_net` command, which only accepts a single interface net.

See Also

- connect_bd_intf_net
- connect_bd_net
- delete_bd_objs
- get_bd_nets
- get_bd_pins
- get_bd_ports

disconnect_bd_net

Disconnect a net from the object.

Syntax

```
disconnect_bd_net [-quiet] [-verbose] <net> <objects>...
```

Returns

TCL_OK, TCL_ERROR if failed.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<net>	The Net that the objects connect to
<objects>	The objects to disconnect from the net

Categories

IPIntegrator

Description

Disconnect a single net in the IP integrator subsystem design from the specified objects.

This command lets you disconnect the specified nets from pins or ports in the IP subsystem design, without deleting the whole net. To delete the whole net, you should use the `delete_bd_objs` command.

This command returns TCL_OK if it is successful, or TCL_ERROR if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Send Feedback

Note: Message limits can be defined with the `set_msg_config` command.

`<net>` - (Required) Specifies a single net in the IP subsystem design to disconnect from the specified objects. The net can be specified by name, or as a single object returned by the `get_bd_nets` command.

`<objects>` - (Required) The list of pin or port objects to disconnect the net from. The pins and ports must be specified as design objects returned by the `get_bd_pins` or `get_bd_ports` commands. They cannot simply be referenced by name.

Example

The following example disconnects the net, here specified by name, from the specified pin objects:

```
disconnect_bd_net /vidout1_locked [get_bd_pins {vidOut1/locked newMod1/t1}]
```

See Also

- connect_bd_intf_net
- connect_bd_net
- delete_bd_objs
- disconnect_bd_intf_net
- get_bd_nets
- get_bd_pins
- get_bd_ports

Send Feedback

disconnect_debug_port

Disconnect nets and pins from debug port channels

Syntax

```
disconnect_debug_port [-channel_index <arg>] [-quiet] [-verbose] <port>
```

Usage

Name	Description
[-channel_index]	Disconnect the net at channel index
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<port>	Debug port name

Categories

Debug

Description

Disconnect signals from the debug ports.

Signals from the Netlist Design are connected to ports of a ILA debug core using the `connect_debug_port` command.

A port can also be deleted from the debug core rather than simply disconnected by using the `delete_debug_port` command.

If you need to determine the specific name of a port on a debug core, use the `get_debug_ports` command to list all ports on a core. You can also use the `report_debug_core` command to list all of the cores in the projects, and their specific parameters.

Arguments

`-channel_index <value>` - (Optional) The channel index of the port to disconnect.

Note: The entire port is disconnected if `-channel_index` is not specified.

`<port>` - (Required) The name of the port on the debug core to disconnect. The port name must be specified as core_name/port_name. See the examples below.

Send Feedback

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example disconnects only the specified channel index from the PROBE1 port of myCore:

```
disconnect_debug_port -channel_index 2 myCore/PROBE1
```

If you do not specify the channel_index, all of the channels of the specified port will be disconnected, as in the following example:

```
disconnect_debug_port myCore/PROBE1
```

See Also

- connect_debug_port
- delete_debug_port
- get_debug_ports
- report_debug_core

Send Feedback

disconnect_hw_server

Close a connection to a hardware server

Syntax

```
disconnect_hw_server [-quiet] [-verbose] [<hw_server>]
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<hw_server>]	hardware server Default: current hardware server

Categories

Hardware

Description

Disconnect the current or specified Vivado tools hardware server from the Vivado Design Suite.

The current hardware server is either the last connected hardware server, or one you have manually defined with the `current_hw_server` command. If you disconnect the current hardware server, there will be no defined current hardware server until you define a new current hw_server object.

This command returns nothing if successful, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_server> - (Optional) The hw_server to disconnect from the Vivado Design Suite. If the server is not specified, the `current_hw_server` will be disconnected. The hardware server must be specified as a hw_server object returned by the `get_hw_servers` or `current_hw_server` commands.

Example

The following example disconnects the specified hardware server from the Vivado Design Suite:

```
disconnect_hw_server [get_hw_server picasso]
```

See Also

- connect_hw_server
- current_hw_server
- get_hw_servers
- refresh_hw_server

Send Feedback

disconnect_net

Disconnect a net from pins or ports

Syntax

```
disconnect_net [-prune] [-net <arg>] [-objects <args>] [-pinlist <args>]
     [-quiet] [-verbose]
```

Usage

Name	Description
[-prune]	When performing disconnect, remove the net and any pin/net chain up to the pin on any primitive instance as long as each object in the chain has only 1 remaining connection.
[-net]	Net to disconnect - optional, net attached to first pin or port object is used if not specified.
[-objects]	List of pin and port names to disconnect. String expressions are supported.
[-pinlist]	List of pin and port objects to disconnect.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Netlist](#)

Description

This command allows the user to disconnect a specified net from one or more pins or ports in the netlist of an open Synthesized or Implemented Design.

Netlist editing changes the in-memory view of the netlist in the current design. It does not change the files in the source fileset, or change the persistent design on the disk. Changes made to the netlist may be saved to a design checkpoint using the `write_checkpoint` command, or may be exported to a netlist file such as Verilog, VHDL, or EDIF, using the appropriate `write_*` command.

Note: Netlist editing is not allowed on the elaborated RTL design.

Arguments

`-prune` - (Optional) Remove hierarchical pins, ports, or nets that are left unconnected after disconnecting the specified nets. This will not remove the pins specified by `-objects`, which are disconnected, but removes the net and pins or ports connected to the specified pins.

Send Feedback

`-net <arg>` - (Optional) Specifies the net to disconnect. If no net is specified, the net connected to the first pin or port object will be disconnected.

Note: Although you can create a bus using the `-bus_from` and `-bus_to` arguments of the `create_net` command, you must disconnect each bit of the bus separately using the `disconnect_net` command.

`-objects <args>` - (Optional) The string list of pin and port names to disconnect from the specified net. String expressions representing patterns are supported. When first-class objects are available, the command line option `-pinlist` should be used instead as it will be faster

`-pinlist <args>` - (Optional) Recommended option over `-objects`. List of pin and port objects to disconnect from the specified net. If the list of pins and ports is built around first-class objects, the -pinlist processing is much faster than `-objects`. This option will also take a list of object names as string, but at the cost of increased runtime.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Example

Using the following connection network:

```
leaf_cell1/pin1 > net1 > block1/pin1 >
    topnet
< block2/pin1 < net2 < leaf_cell2/pin1
```

This example first disconnects the signal, topnet, from block1/pin1, and then prunes topnet, block2/pin1, and net2, but does not prune leaf_cell2/pin1:

```
disconnect_net -prune -net topnet -objects [get_pins block1/pin1]
```

Note: net2 is not removed, because block1/pin1 is not pruned as part of the `-prune` option.

The following example disconnects the specified bit of the dataBus:

```
disconnect_net -net dataBus[1] -objects {dataIN[1] myDMA/data[1]}
```

This example demonstrates the most efficient coding style to disconnect nets from multiple pins or ports. In the first code sample, all pins are presented as a single list of pins or ports, and the `disconnect_net` command runs more quickly than the second code sample that is presented as multiple `disconnect_net` commands:

```
disconnect_net -objects [list {pad_rin1_IBUF[0]_inst/O} {hcore0/pad_rin1[9]
[0]} \
{pad_rin1_IBUF[1]_inst/O} {hcore0/pad_rin1[9][1]} \
{pad_rin1_IBUF[2]_inst/O} {hcore0/pad_rin1[9][2]} \
{pad_rin1_IBUF[3]_inst/O} {hcore0/pad_rin1[9][3]} \
{pad_rin1_IBUF[4]_inst/O} {hcore0/pad_rin1[9][4]} \
{pad_rin1_IBUF[5]_inst/O} {hcore0/pad_rin1[9][5]} \
{pad_rin1_IBUF[6]_inst/O} {hcore0/pad_rin1[9][6]} \
{pad_rin1_IBUF[7]_inst/O} {hcore0/pad_rin1[9][7]} \
{pad_rin1_IBUF[8]_inst/O} {hcore0/pad_rin1[9][8]} \
{pad_rin1_IBUF[9]_inst/O} {hcore0/pad_rin1[9][9]} ]
```

TIP: *The lack of the `-net` option in these samples means that the net attached to the first pin or port specified is the net that will be disconnected from all subsequent pins and ports.*

```
disconnect_net -objects [list {pad_rin1_IBUF[0]_inst/O} {hcore0/pad_rin1[9]
[0]}]
disconnect_net -objects [list {pad_rin1_IBUF[1]_inst/O} {hcore0/pad_rin1[9]
[1]}]
disconnect_net -objects [list {pad_rin1_IBUF[2]_inst/O} {hcore0/pad_rin1[9]
[2]}]
disconnect_net -objects [list {pad_rin1_IBUF[3]_inst/O} {hcore0/pad_rin1[9]
[3]}]
disconnect_net -objects [list {pad_rin1_IBUF[4]_inst/O} {hcore0/pad_rin1[9]
[4]}]
disconnect_net -objects [list {pad_rin1_IBUF[5]_inst/O} {hcore0/pad_rin1[9]
[5]}]
disconnect_net -objects [list {pad_rin1_IBUF[6]_inst/O} {hcore0/pad_rin1[9]
[6]}]
disconnect_net -objects [list {pad_rin1_IBUF[7]_inst/O} {hcore0/pad_rin1[9]
[7]}]
disconnect_net -objects [list {pad_rin1_IBUF[8]_inst/O} {hcore0/pad_rin1[9]
[8]}]
disconnect_net -objects [list {pad_rin1_IBUF[9]_inst/O} {hcore0/pad_rin1[9]
[9]}]
```

See Also

- connect_net
- remove_net
- create_net
- write_checkpoint
- write_edif
- write_verilog
- write_vhdl

display_hw_ila_data

Display hardware ILA data in viewer

Syntax

```
display_hw_ila_data [-wcfg <arg>] [-reset] [-quiet] [-verbose]
    [<hw_ila_data>...]
```

Usage

Name	Description
[-wcfg]	Import alternate wave config file
[-reset]	Force reset wave config file to default configuration
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<hw_ila_data>]	List of hardware ILA data objects. Default: Current hardware ILA data

Categories

[Hardware](#)

Description

This command is intended for use with the graphical user interface of the Vivado Design Suite logic analyzer feature. It displays the specified ILA debug core data object in a wave config window of the Vivado logic analyzer.

The ILA debug sample data is acquired from a running device using the `upload_hw_ila_data` command. This creates a hw_ila_data object that can be written to a file on disk using the `write_hw_ila_data` command. This command reads that ILA data file.

The display characteristics of the ILA debug core in the waveform window are determined by the Wave Config file. The Wave Config file contains just the list of wave objects (signals, dividers, groups, virtual buses) to display, and their display properties, plus markers.

A wave configuration object is created in the Vivado logic analyzer with the `create_wave_config` command. A Wave Config file is written to disk by the use of the `save_wave_config` command, and can be opened with the `open_wave_config` command.

The `open_wave_config` command opens a Wave Config file and maps it to the data source in the current simulation.

Send Feedback

Arguments

`-wcfg <arg>` - (Optional) View the ILA data using the specified Wave Config file, created using the `save_wave_config` command. If the `-wcfg` option is not specified, the ILA data will display in a default wave configuration as determined by the Vivado logic analyzer.

`-reset` - (Optional) Reset the waveform window to the default configuration and display the specified ILA data.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_ila_data> - (Optional) Specify one or more hw_ila_data objects to display. The hw_ila_data must be specified as an object as returned by the `get_hw_ila_datas` or `current_hw_ila_data` commands. If the hardware ILA data is not specified, the `current_hw_ila_data object` will be displayed.

Example

The following example reads a hw_ila_data file, and displays the resulting hw_ila_data object in the waveform window of the Vivado logic analyzer:

```
read_hw_ila_data C:/hw_ila_file.ila
display_hw_ila_data [get_hw_ila_datas hw_ila_file]
```

See Also

- current_hw_ila
- current_hw_ila_data
- upload_hw_ila_data
- get_hw_ilas
- get_hw_ila_datas
- read_hw_ila_data
- run_hw_ila
- save_wave_config

Send Feedback

display_hw_sio_scan

Display an existing hardware SIO scan.

Syntax

```
display_hw_sio_scan [-quiet] [-verbose] <hw_sio_scans>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hw_sio_scans>	hardware SIO scans

Categories

Hardware

Description

This command is intended for use with the graphical user interface of the Vivado Design Suite serial I/O analyzer feature. It displays the specified SIO scan data object, or objects, in a Scan Plots window of the Vivado IDE.

The SIO scan data can be read from a file on disk using the `read_hw_sio_scan` command, or from a hw_sio_scan object created by the `run_hw_sio_scan` command. The type of plot displayed is determined by the <scan_type> of the hw_sio_scan object.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_sio_scans> - (Optional) Specify one or more existing hw_sio_scan objects to display in Scan Plot windows in the Hardware Manager feature of the Vivado IDE.

Send Feedback

Example

The following example reads an SIO scan data file into memory, and displays the hw_sio_scan object that is created:

```
display_hw_sio_scan [read_hw_sio_scan C:/Data/loopback1.csv]
```

See Also

- create_hw_sio_scan
- current_hw_device
- get_hw_sio_scans
- remove_hw_sio_scan
- run_hw_sio_scan
- read_hw_sio_scan
- stop_hw_sio_scan
- wait_on_hw_sio_scan
- write_hw_sio_scan

encrypt

Encrypt files in place with a language specific key file in IEEE 1735. no default

Syntax

```
encrypt [-key <arg>] -lang <arg> [-ext <arg>] [-quiet] [-verbose]
    <files>...
```

Usage

Name	Description
[-key]	key file to be used to encrypt; if absent, use embedded keys
-lang	HDL language of the input/output file
[-ext]	extension to use for encrypted file; the original source files will be preserved.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<files>	Files to be encrypted in place

Categories

FileIO

Description

> 💡 **TIP:** *The* `encrypt` *command is provided with limited access, and requires a special license to use.*

Allows anyone with an encryption license to encrypt Verilog or VHDL files using the IEEE 1735 encryption standard.

Encrypted files can be provided by third-party IP providers to protect their intellectual property, while still enabling the Vivado Design Suite to read the encrypted files for synthesis and simulation. The data is in plain text prior to encryption.

> ⭐ **IMPORTANT!** *Unless the* `-ext` *option is used, the specified files are encrypted in place, overwriting the input files with the encrypted files.*

Arguments

`-key <arg>` - (Optional) Specifies an RSA key file that includes the Xilinx public key. If the `-key` is not specified, the Vivado tool looks for keys embedded within the specified files. These are 1735 supported pragmas, or directives embedded into the specified files, that provide the encryption key and indicate where the protected data begins and ends.

> 💡 **TIP:** *The Xilinx public key can be obtained by members from the IEEE P1735 working group, or by contacting an appropriate Xilinx representative.*

`-lang [ vhdl | verilog ]` - (Required) Specify the HDL language of the source files to be encrypted. Supported values are VHDL or verilog.

`-ext <arg>` - (Optional) Specify an extension to use for the output encrypted files. The original source files will be preserved.

> ⭐ **IMPORTANT!** *If this option is not specified, the original source files will be overwritten with the encrypted output.*

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<files>` - (Required) The names of Verilog or VHDL files to encrypt.

Example

The following example encrypts the specified Verilog file, using the specified key file:

```
encrypt -lang verilog -key C:/Data/xilinx_rsa_key.txt C:/Data/design_1.v
```

Note: The specified source file is overwritten by the encrypted output file.

See Also

- write_verilog
- write_vhdl

endgroup

End a set of commands that can be undone/redone as a group

Syntax

```
endgroup [-quiet] [-verbose]
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[GUIControl](#)

Description

Ends a sequence of commands that can be undone or redone as a series. Use `startgroup` to start the sequence of commands.

You can have multiple command groups to `undo` or `redo`, but you cannot nest command groups. You must use `endgroup` to end a command sequence before using `startgroup` to create a new command sequence.

> 💡 **TIP:** *The `startgroup/endgroup` commands are provided to support sequences of related commands that can be undone via the `undo` command, or redone if needed using the `redo` command. However, some Tcl commands can trigger an `endgroup` unexpectedly, and certain commands do not support either `UNDO` or `REDO`. The limitations are not fully defined.*

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example defines a `startgroup`, executes a sequence of related commands, and then executes the `endgroup`. This sequence of commands can be undone as a group:

```
startgroup
create_pblock pblock_wbArbEngine
create_pblock pblock_usbEng
add_cells_to_pblock pblock_wbArbEngine \
    [get_cells [list wbArbEngine]] -clear_locs
add_cells_to_pblock pblock_usbEng \
    [get_cells [list usbEngine1/usbEngineSRAM]] -clear_locs
endgroup
```

See Also

- startgroup
- redo
- undo

Send Feedback

exclude_bd_addr_seg

Exclude segment from an address space.

Syntax

```
exclude_bd_addr_seg [-target_address_space <arg>] [-quiet] [-verbose]
    [<segment_to_exclude>]
```

Returns

The newly excluded segment object, "" if failed.

Usage

Name	Description
[-target_address_space]	Target address space to exclude the slave segment from
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<segment_to_exclude>]	segment to exclude

Categories

[IPIntegrator](#)

Description

Exclude the specified AXI peripheral address segment from access by the AXI master it is mapped to, in order to support sparse connectivity and eliminate unneeded device resources.

This command lets you exclude specific peripherals from being accessed by specific AXI masters. For example, in the case where AXI peripherals P0 and P1 are connected to two masters M0 and M1, you can use sparse connectivity to let M0 access both P0 and P1, and let M1 accesses P1, but exclude it from P0.

In the IP integrator block design, address segments of AXI peripherals will have one of three states:

- Unmapped - An AXI peripheral, or slave interface, is connected to an AXI master, but the peripheral has not been assigned an address segment in the master's address space and is not visible to the master.

- Mapped - The AXI peripheral is mapped into the AXI master's address space, assigned an address segment or range, and is accessible through the master.

Send Feedback

Chapter 3: Tcl Commands Listed Alphabetically

- Excluded - The AXI peripheral is mapped to the AXI master, and has been assigned an address, but is not accessible to the master. The address segment that the AXI slave occupies within the master address space is also considered filled.

The purpose of excluding the address segment is to restrict access to peripherals that are connected to multiple masters. The `validate_bd_design` command will return a critical warning if a peripheral interface is connected to a master, but not mapped to an address segment of that master. However, by excluding the peripheral after it is mapped, the resources required to connect and provide access between the AXI master and the peripheral (the muxes and decoding for example) can be eliminated to conserve resources on the implemented design.

> **TIP:** *When running assign_bd_address, the IP integrator will map unmapped address segments into address spaces, but will not map excluded address spaces.*

This command offers two syntaxes, for a previously mapped address segment, and an unmapped address segment:

```
exclude_bd_addr_seg <master_addr_seg>
exclude_bd_addr_seg -target_address_space <master_addr_space>
<slave_addr_seg>
```

In the second command syntax, when a slave segment is specified, the slave will first be assigned or mapped to the specified AXI master address space, and then it will be excluded from access by the master.

This command returns nothing if successful, or returns an error if it failed.

Arguments

`-target_address_space <arg>` - (Optional) The target address space to map the address segment into. This option provides the mapping assignment usually performed by the `assign_bd_address` command, to let you assign the address segment to an address space, and exclude the address segment in a single command.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<segment_to_exclude> - Specify a single address segment object, bd_addr_seg, to exclude from access by its assigned AXI master.

UG835 (v2021.1) June 16, 2021
Tcl Command Reference Guide

Send Feedback

www.xilinx.com
509

Example

The following example assigns an address segment defined by an AXI peripheral, or slave, to an AXI master address space, and then excludes the address segment from access by that master:

```
assign_bd_address [get_bd_addr_segs {axi_gpio_1/S_AXI/Reg }]
exclude_bd_addr_seg [get_bd_addr_segs microblaze_1/Data/SEG_axi_gpio_1_Reg]
```

See Also

- assign_bd_address
- create_bd_addr_seg
- get_bd_addr_segs
- get_bd_addr_spaces
- include_bd_addr_seg

Send Feedback

execute_hw_svf

Execute SVF file on current_hw_target

Syntax

```
execute_hw_svf [-quiet] [-verbose] <file_name>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<file_name>	SVF filename

Categories

Hardware

Description

The Vivado hardware manager supports programming of hardware devices through the use of Serial Vector Format (SVF) files. SVF files are ASCII files that contain both programming instructions and configuration data. These files are used by ATE machines and embedded controllers to perform boundary-scan operations. The SVF file is an ASCII files that captures the JTAG commands needed to program the bitstream directly into a Xilinx device, or indirectly into a flash memory device. The SVF file can be written using the `write_hw_svf` command, or used to program a device through the `execute_hw_svf` command. Refer to the *Vivado Design Suite User Guide: Programming and Debugging* (UG908) for more information.

The `execute_hw_svf` command converts the SVF commands into Vivado Tcl commands and executes them on the specified target. This process could take some time depending on how big the SVF file is. The command requires an open, current hw_target object, with a JTAG chain that matches the device chain specified in the SVF file.

> **TIP:** *The `execute_hw_svf` command is not a general purpose SVF reader, and should only be used to read and execute SVF files written by the Vivado tools.*

This command returns a transcript of its process, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Send Feedback

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from the `execute_hw_svf` command. Use this option when you are debugging problems related to executing the SVF file, as this option will display all of the pseudo-SVF commands that the Vivado tool is running while executing the file.

`<file_name>` - Specifies the SVF file name to execute.

Note: If the path is not specified as part of the file name, the tool will search for the specified file in the current working directory and then in the directory from which the tool was launched.

Examples

The following example executes the specified SVF command file in verbose mode to display all of the commands being run:

```
open_hw_target {houdini26:3121/xilinx_tcf/Digilent/210203327996A}
execute_hw_svf -verbose C:/Data/k7_design.svf
```

See Also

- create_hw_device
- create_hw_target
- current_hw_target
- current_project
- get_hw_probes
- open_hw_target
- program_hw_devices
- write_hw_svf

export_as_example_design

Export current design as a static example design

Syntax

```
export_as_example_design -vlnv <arg> [-no_ip_version] [-force] [-quiet]
    [-verbose] -directory <arg>
```

Usage

Name	Description
-vlnv	VLNV of the example design to be generated
[-no_ip_version]	Flag to not include the IP version as part of the IP VLNV in create_bd_cell commands.
[-force]	create a directory if it does not exist
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
-directory	Destination directory in which example design needs to be generated

Categories

IPIntegrator

export_bd_synth

(User-written application) Create and write a single design checkpoint and stub files for a Block Design (BD), for use with third party synthesis tools. Perform synthesis as necessary.

Syntax

```
export_bd_synth [-force] [-keep] [-verbose] [-quiet] <file>
```

Returns

(none) An error will be thrown if the command is not successful.

Usage

Name	Description
[-force]	Overwrite existing design checkpoint and stub files
[-keep]	Keep the temporary directory and project
[-verbose]	Print verbose messaging
[-quiet]	Ignore command errors
<file>	The Block Design file to write a synthesized checkpoint for

Categories

[xilinxtclstore](), [projutils]()

Description

Runs synthesis for a block design (`.bd`), integrates the design along with any needed sub-designs (e.g. out-of-context synthesized IP), and writes out a single design checkpoint (`.dcp`) of the entire synthesized design, as well as HDL stub files, for use in other synthesis tools. The output files will be placed in the same directory as the source BD file.

Arguments

`-force` - (Optional) Overwrite any existing design checkpoint and stub files of the same name.

`-keep` - (Optional) Keep the temporary directory and project after `export_bd_synth` has finished.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

Send Feedback

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file>` - (Required) The block design file (.bd) from which to export a synthesized checkpoint.

Examples

The following command will generate a synthesis checkpoint, with stub files, for the specified block design:

```
export_bd_synth [get_files block_1.bd]
```

See Also

- get_files

export_ip_user_files

(User-written application) Generate and export IP/IPI user files from a project. This can be scoped to work on one or more IPs.

Syntax

```
export_ip_user_files [-of_objects <arg>] [-ip_user_files_dir <arg>]
    [-ipstatic_source_dir <arg>] [-lib_map_path <arg>] [-no_script] [-sync]
    [-reset] [-force] [-quiet] [-verbose]
```

Returns

List of files that were exported.

Usage

Name	Description
`[-of_objects]`	IP,IPI or a fileset object Default: None
`[-ip_user_files_dir]`	Directory path to simulation base directory (for static, dynamic, wrapper, netlist, script and MEM files) Default: None
`[-ipstatic_source_dir]`	Directory path to the IP static files Default: None
`[-lib_map_path]`	Compiled simulation library directory path Default: Empty
`[-no_script]`	Do not export simulation scripts Default: 1
`[-sync]`	Delete IP/IPI dynamic and simulation script files
`[-reset]`	Delete all IP/IPI static, dynamic and simulation script files
`[-force]`	Overwrite files
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution

Categories

xilinxtclstore

Description

Export IP user files repository with static, dynamic, netlist, verilog/vhdl stubs and memory initializaton files.

Arguments

`-of_objects <arg>` - (Optional) Specify the target object, IP, block design (`.bd`) or a fileset, for which the IP static and dynamic files need to be exported.

Send Feedback

`-ip_user_files_dir <arg>` - (Optional) Directory path to IP user files base directory (for dynamic and other IP non-static files). By default, if this switch is not specified then this command will use the path specified with the "IP.USER_FILES_DIR" project property value.

`-ipstatic_source_dir <arg>` - (Optional) Directory path to the static IP files. By default, if this switch is not specified then this command will use the path specified with the "SIM.IPSTATIC_SOURCE_DIR" project property value.

Note: If the `-ip_user_files_dir` option is specified, by default the IP static files will be exported under the sub-directory with the name "ipstatic". However, if this option is also specified, then the IP static files will be exported in the path specified with this option.

`-lib_map_path <arg>` - (Optional) Compiled simulation library directory path.

`-no_script` - (Optional) Do not export simulation scripts.

`-sync` - (Optional) Delete IP/IPI dynamic and simulation script files.

`-reset` - (Optional) Delete all IP/IPI static, dynamic and simulation script files.

`-force` - (Optional) Overwrite existing files.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following command will export the dynamic files for the `char_fifo` IP to `<project>/<project>.ip_user_files/ip/char_fifo` directory, and IP static files to `<project>/<project>.ip_user_files/ipstatic` directory:

```
export_ip_user_files -of_objects [get_ips char_fifo]
```

See Also

- get_files
- get_ips

export_simulation

(User-written application) Export a script and associated data files (if any) for driving standalone simulation using the specified simulator.

Syntax

```
export_simulation [-simulator <arg>] [-of_objects <arg>]
    [-ip_user_files_dir <arg>] [-ipstatic_source_dir <arg>]
    [-lib_map_path <arg>] [-gcc_install_path <arg>] [-script_name <arg>]
    [-directory <arg>] [-runtime <arg>] [-define <arg>] [-generic <arg>]
    [-include <arg>] [-more_options <arg>] [-use_ip_compiled_libs]
    [-absolute_path] [-export_source_files] [-generate_hier_access]
    [-32bit] [-force] [-quiet] [-verbose]
```

Returns

None

Usage

Name	Description
[-simulator]	Simulator for which the simulation script will be created (value=all|xsim|modelsim|questa|ies|xceliu m|vcs| riviera|activehdl) Default: all
[-of_objects]	Export simulation script for the specified object Default: None
[-ip_user_files_dir]	Directory path to the exported IP/BD (Block Design) user files (for static, dynamic and data files) Default: Empty
[-ipstatic_source_dir]	Directory path to the exported IP/BD static files Default: Empty
[-lib_map_path]	Precompiled simulation library directory path. If not specified, then please follow the instructions in the generated script header to manually provide the simulation library mapping information. Default: Empty
[-gcc_install_path]	GNU compiler installation directory path for the g++/gcc executables. Default: Empty
[-script_name]	Output script filename. If not specified, then a file with a default name will be created. Default: top_module.sh
[-directory]	Directory where the simulation script will be generated Default: export_sim
[-runtime]	Run simulation for this time (default:full simulation run or until a logical break or finish condition) Default: Empty
[-define]	Read verilog defines from the list specified with this switch Default: Empty
[-generic]	Read vhdl generics from the list specified with this switch Default: Empty
[-include]	Read include directory paths from the list specified with this switch Default: Empty
[-more_options]	Pass specified options to the simulator tool Default: Empty

Send Feedback

Name	Description
`[-use_ip_compiled_libs]`	Reference pre-compiled IP static library during compilation. This switch requires -ip_user_files_dir and -ipstatic_source_dir switches as well for generating scripts using pre-compiled IP library.
`[-absolute_path]`	Make all file paths absolute
`[-export_source_files]`	Copy IP/BD design files to output directory
`[-generate_hier_access]`	Extract path for hierarchical access simulation
`[-32bit]`	Perform 32bit compilation
`[-force]`	Overwrite previous files
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution

Categories

xilinxtclstore

Description

Export a simulation script file for the target simulator. The generated script will contain simulator commands for compiling, elaborating, and simulating the design.

This command retrieves the simulation compile order of specified objects, and exports this information in a text file with the compiler commands and default options for the target simulator. The specified object can be a simulation fileset or an IP. If the object is not specified, the `export_simulation` command will generate the script for the simulation top.

By default, the design source file and include directory paths in the compiler command line will be set relative to the directory where the script file is generated. To make these paths absolute, specify the `-absolute_path` option.

The command will also copy data files (if any) from the fileset, or from an IP, to the output directory. If the design contains Verilog sources, the generated script will also copy `glbl.v` from the software installation path to the output directory.

Note: In order to perform simulation with the generated script, the simulation libraries must be compiled first using the `compile_simlib` command, with the compiled library directory path specified, when generating this script. The generated simulation script will automatically include the setup files for the target simulator from the compiled library directory. Use the `-lib_map_path` option to specify the compiled library directory.

Note: If a design contains IP/BD sources, these sources must be generated first before exporting the script. `export_simulation` does not generate output products of these sources, but checks for the status of these sources and print message accordingly.

This command returns nothing.

Send Feedback

Arguments

`-simulator <arg>` - (Optional) Specify the target simulator name for which the simulation script needs to be generated. Valid simulator names are `all`, `xsim`, `modelsim`, `questa`, `ies`, `xcelium`, `vcs`, `riviera`, and `activehdl`. If this option is not specified the command will generate the script for all simulators.

`-of_objects <arg>` - (Optional) Specify the target object for which the simulation script file needs to be generated. The target object can be a simulation fileset (simset) or an IP/BD source. If this option is not specified the command will generate simulation scripts for the current simulation fileset.

Note: The `-of_objects` option requires objects to be specified using the `get_files`, `get_ips`, `current_fileset`, or `current_fileset -simset` commands rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-ip_user_files_dir <arg>` - (Optional) Specify the directory path to the exported IP/BD generated source files. This is generally the sub-directory named `<project>.ip_user_files` in the project directory structure.

`-ipstatic_source_dir <arg>` - (Optional) Specify the directory path to the exported IP/BD generated static source files. This is generally the sub-directory named `<project>.ip_user_files/ipstatic` in the project directory structure.

`-lib_map_path <arg>` - (Optional) Specify the pre-compiled simulation library directory path where the Xilinx simulation libraries are compiled.

* To specify a library location if you specified a simulator name with the `-simulator` switchuse the following syntax.

  ```
  -lib_map_path <compiled-library>
  ```

* To specify multiple library locations if you specified the `all` value with the `-simulator` switch, use the following syntax

  ```
  -lib_map_path [list {<simulator>=<compiled-library>}
                {<simulator>=<compiled-library>}
                ...
                {<simulator>=<compiled-library>}]
  ```

  Where `<simulator>` is the name of the simulator and <compiled-library> is the path to the simulation library compiled for the simulator specified with `<simulator>`.

`-gcc_install_path <arg>` - (Optional) Specify the directory path to the GNU executables for the target simulator. This option is required for designs that contain SystemC, C++, or C sources.

`-script_name <arg>` - (Optional) Specify the name of the generated script. If this option is not specified, the filename is generated based on the object type selected using `-of_objects`, and will be in one of the following formats:

- `<simulation_top_name>_sim_<simulator>.sh`

- `<ip_name>_sim_<simulator>.sh`

`-directory <arg>` - (Optional) Specify the directory path where the script file needs to be generated. By default, the script will be generated in the `export_sim/<simulator>` directory.

`-runtime <arg>` - (Optional) Specify the simulation run time. If this option is not specified, then the simulation will run for infinite time or until a logical break or finish condition.

`-define <arg>` - (Optional) Specify the Verilog defines applicable for the current design. The syntax for specifying the defines is:

```
-define [list {<variable[=value]} ... [{<variable[=value]>}]]
```

`-generic <arg>` - (Optional) Specify the VHDL generics applicable for the current design. The syntax for specifying the generic variables is:

```
-generic [list {<variable[=value]} ... [{<variable[=value]>}]]
```

`-include <arg>` - (Optional) Specify the include directory paths to the source files in the design. The syntax for specifying the include file directory paths is:

```
-include [list {<include-directory} ... [{<include-directory>}]]
```

`-more_options <arg>` - (Optional) Specify additional switches/args for the simulator tools that need to be added when executing the script. The syntax for specifying the `switches/ args` is:

```
-more_options [list {<simulator>.<step>.<tool>:<list of switches/args>}
               {<simulator>.<step>.<tool>:<list of switches/args>}
               ...
               {<simulator>.<step>.<tool>:<list of switches/args>}]
```

Where `<simulator>` is the name of the simulator, `<step>` is the name of the step (compile, elaborate, simulate), and `<tool>` is the name of the simulator tool (xvlog, vcom, vcs, etc.).

`-use_ip_compiled_libs` - (Optional) Reference precompiled IP static library during compilation. The IP static source will not be compiled but instead it is compiled version will be referenced during elaboration. This switch requires `-ip_user_files_dir` and the `-ipstatic_source_dir` switches as well.

`-absolute_path` - (Optional) Specify this option to make the source and include directory paths into absolute format in the generated script file. By default, all the paths are written relative to the script output directory specified with the `-directory` option.

`-export_source_files` - (Optional) Specify this option to copy the design, generated IP/BD, and data sources to the `<output directory>/<simulator>/srcs` directory. If `-of_objects` specifies an IP/BD, the sources will be copied to the `<output directory>/<IP/BD>/<simulator>/srcs` directory. The source and include file paths in the generated script will be set relative to the `srcs` directory (the file paths can be made absolute using the `-absolute_path` switch).

`-generate_hier_access` - (Optional) Specify this option to extract the path for hierarchical access simulation.

`-32bit` - (Optional) Specify this option to perform 32-bit simulation. If this option is not specified then by default a 64-bit option will be added to the simulation command line.

`-force` - (Optional) Overwrite an existing script file of the same name. If the script file already exists, the tool returns an error unless the `-force` argument is specified.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

Generate a script file named accum_0_sim_xsim.sh for the XSim simulator in test_sim directory.

```
export_simulation -simulator xsim -of_objects [get_ips accum_0] \
    -ip_user_files_dir <path> \
    -ipstatic_source_dir <path> \
    -directory test_sim
```

Generate a script file named `test.sh` for Questa simulator using the compiled simulation library from `/test/questa/clibs`. The command will also copy the `accum_0` IP source files into the `./results/accum_0/questa/srcs` directory.

```
export_simulation -simulator questa -of_objects [get_ips accum_0] \
-lib_map_path /test/questa/clibs \
-runtime 2000ns \
-define [list {FREQ=10} {RATE=20}] \
-generic [list {DATA=34} {LINE=21}] \
-script_name test.sh \
-directory ./results \
-export_source_files
```

Generate a script file named verify.sh for the Xcelium simulator for the current simulation fileset.

```
export_simulation -simulator xcelium -of_objects [current_fileset -simset] \
    -lib_map_path /test/xcelium/clibs \
    -include [list {/test/fifo/ver/incl_1} {/test/fifo/ver/incl_2}] \
    -script_name verify.sh \
    -directory ./results
```

Generate and execute the script for the VCS simulator:

1. Export the script for the `char_fifo` IP.

```
export_simulation -simulator vcs -of_objects [get_ips char_fifo] \
    -lib_map_path /test/vcs/clibs \
    -ip_user_files_dir <path> \
    -ipstatic_source_dir <path> \
    -runtime 5000ns \
    -more_options [list {vcs.compile.vlogan:-V} {vcs.elaborate.vcs:-
timescale 1ps/1ps}] \
    -script_name verify.sh \
    -directory ./xp \
    -use_ip_compiled_libs
    -absolute_path \
    -force
```

2. Verify the simulation for the `char_fifo` IP.

```
cd xp/char_fifo/vcs
    ./verify.sh
```

See Also

- compile_simlib
- current_fileset
- get_files
- get_ips

extract_files

Extract files from a core container to disk

Syntax

```
extract_files [-base_dir <arg>] [-force] [-no_ip_dir] [-no_paths] [-quiet]
    [-verbose] <files>...
```

Returns

List of files that were extracted with the new paths.

Usage

Name	Description
[-base_dir]	Base directory for extracted files Default: ip_files
[-force]	Overwrite existing files
[-no_ip_dir]	Don't include the IP dir as part of the extract directory
[-no_paths]	Don't include directories when extracting files
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<files>	Name of the file(s) to be extracted

Categories

IPFlow

Description

Extract the files from an IP in core container format.

The core container format for IP is a compressed zip file that reduces the file structure in the design, and increases tool performance.

This command returns a list of files extracted from the core container IP, or returns an error if it fails.

Arguments

-base_dir <arg> - (Optional) Specify the directory to write the extracted files into. By default the extract_files command will write files into a folder called ip_files inside of the current working directory.

-force - (Optional) Overwrite existing files of the same name if any exist.

`-no_ip_dir` - (Optional) Don't include an IP sub-folder as part of path for the the extracted files. In this case, the files will be exported to the specified directory, without a sub-folder named after the core container IP.

`-no_paths` - (Optional) Don't include sub-folders of the core container in the extracted files. This option will cause all files to be extracted to the top-level `ip_files` folder, or the folder specified by the `-base_dir` option.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<files>` - (Required) Specify the name of the IP core container (`.XCIX`) file to extract the files from.

Examples

The following example extracts the files from the specified core container format IP to the specified base directory:

```
extract_files -base_dir C:/Data [get_files char_fifo.xcix]
```

See Also

- convert_ips
- create_ip
- get_files
- get_ips

filter

Filter a list, resulting in new list

Syntax

```
filter [-regexp] [-nocase] [-quiet] [-verbose] [<objects>] [<filter>]
```

Returns

New list

Usage

Name	Description
[-regexp]	Operators =~ and !~ use regular expressions
[-nocase]	Perform case-insensitive matching (valid only when -regexp specified)
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<objects>]	List of objects to filter
[<filter>]	Filter list with expression

Categories

Object, PropertyAndParameter, XDC

Description

Takes a list of objects, and returns a reduced list of objects that match the specified filter search pattern.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<objects>` - (Optional) A list of objects that should be filtered to reduce the set to the desired results. The list of objects can be obtained by using one of the many `get_*` commands such as `get_parts`.

<filter> - (Optional) The expression to use for filtering. The specified pattern filters the list of objects returned based on property values on the objects. You can find out which properties are on an object with the `report_property` or `list_property` command. Any property/value pair can be used as a filter. For example, in the case of the "part" object, "DEVICE", "FAMILY" and "SPEED" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

Examples

The following example returns a list of parts filtered for the specified speed grade:

```
filter [get_parts] {speed == -3}
```

The following example filters parts based according to speed grade -3 OR speed grade -2. All parts matching either speed grade will be returned.

```
filter [get_parts] {speed == -3  ||  speed == -2}
```

The following example uses regular expression and returns a list of VStatus ports in the design, with zero or more wildcards, and the numbers 0 to 9 appearing one or more times within square brackets:

```
filter -regexp [get_ports] {NAME =~ VStatus.*\[[0-9]+\]}
```

See Also

- get_parts
- get_ports

Send Feedback

find_bd_objs

Find a list of pins, ports or interfaces with a given relationship to the given object.

Syntax

```
find_bd_objs -relation <arg> [-boundary_type <arg>] [-thru_hier]
    [-stop_at_interconnect] [-stop_at_container] [-end_type <arg>] [-quiet]
    [-verbose] <objects>...
```

Returns

List of pins, ports or interface objects, "" if failed.

Usage

Name	Description
-relation	Relation to the input objs: connected_to, addressable_slave, addressing_master. 'connected_to' will find corresponding pins, ports or interfaces that are connected to the given source objects, across hierarchy boundaries.
[-boundary_type]	Used when source object is an hierarchical block's pin or interface pin. Valid values are empty string for same level (default), 'lower', or 'all'. If 'lower' boundary, searches from within hierarchy. This option is only valid for relation: connected_to
[-thru_hier]	Flag used to ignore boundary of hierarchical blocks. If used used with boundary_type 'lower', flag will only affect the hierarchical blocks within parent hierarchical block.
[-stop_at_interconnect]	Flag used to stop at the axi_interconnect's boundary when -thru_hier is used.
[-stop_at_container]	Flag used to stop at the container's boundary when -thru_hier is used.
[-end_type]	Only to be used with objects that are pins or ports and bus interface pins or ports. For pins/ports - Default is to return the sink objects for a given source object and to return the source object for a given sink object. If 'all' is used for a given sink object, will return both source and other sink objectst that are connected to the source object. For bus interface pins/ports - Default is to return the end connection that is non-monitor interfaces. If 'monitor' is used, will only return the monitor interfaces. If 'all' is used, will return both end connection and monitor interfaces. If 'master', will return master interfaces and if 'slave', will return slave interfaces.This option is only valid for relation: connected_to
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<objects>	One or more source object to start finding from

Categories

IPIntegrator

Send Feedback

Description

Find a list of pins, ports or interfaces with a given relationship to the specified objects.

This command returns a list of pins, ports or interface objects if successful, or returns an error if it fails.

Arguments

`-relation <arg>` - (Required) Specifies the relationship of the objects to find with the objects specified to seed the search. Valid values are:

- `connected_to` - Find corresponding pins, ports or interfaces that are connected to the given source objects. The search occurs within the same level of hierarchy as the specified objects. Use the `-thru_hier` option to search across hierarchical boundaries.

- `addressable_slave` - Find pins, ports, or interfaces, that can be addressed as slaves by the specified search objects.

- `addressing_master` - Find pins, ports, or interfaces, that address the specified search objects as masters.

`-boundary_type [ upper | lower | all ]` - (Optional) Used when the specified source objects include a pin or interface pin on a hierarchical module. The default behavior is "upper", which searches for related objects within the same level as the hierarchical module that the pin is found on. If "lower", the search occurs inside the hierarchical module that the pin is on. If "all", then both the level of the hierarchical module, and inside the hierarchical module are searched for related objects.

Note: This option must be used with `-relation connected_to`.

`-thru_hier` - (Optional) Ignore the boundary of hierarchical blocks when performing the search for objects. If used used with `-boundary_type lower`, the search for related objects will start inside the hierarchical module the pin is on, and continue searching downward through the hierarchy.

`-stop_at_interconnect` - (Optional) Specifies to stop searching at the boundary of the AXI Interconnect IP when `-thru_hier` is specified. Other hierarchical blocks are searched as usual.

`-end_type all` - (Optional) This option can be specified for bd_pin and bd_port type `<objects>`. The default search for related objects returns all of the sink pins attached to a specified source object, and to return the source object for a given sink. When `-end_type all` is specified for a sink object, the tool will return the source pin or port of the specified sink object, as well as other sink objects that are connected to the source object.

Note: This option must be used with `-relation connected_to`.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<objects>` - (Required) One or more objects to use to find related objects.

Example

The following example gets pins, ports, and interfaces connected to the specified object, across all levels of the block design hierarchy:

```
find_bd_objs -relation connected_to -thru_hier \
[get_bd_pins /proc_sys_reset_1/peripheral_aresetn]
```

See Also

- create_bd_cell
- create_bd_intf_net
- create_bd_intf_pin
- create_bd_intf_port
- create_bd_net
- create_bd_pin
- create_bd_port

find_routing_path

Find a routing path between two nodes.

Syntax

```
find_routing_path [-ignore_all_routing] [-allow_overlap] [-max_nodes <arg>]
    [-min_nodes <arg>] [-from <args>] [-to <args>] [-quiet] [-verbose]
```

Usage

Name	Description
[-ignore_all_routing]	Solution may include nodes used in any existing (including fixed) routes.
[-allow_overlap]	Solution may include nodes used in existing non-fixed routes.
[-max_nodes]	Specifies the maximum number of nodes (including from and to nodes) allowed in solution. Default: 100
[-min_nodes]	Specifies the minimum number of nodes (including from and to nodes) allowed in solution. Default: 2
[-from]	-from <start node> Start of routing path.
[-to]	-to <end node> End of routing path.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Object](#)

Description

Finds a routing solution between two nodes on an unrouted, or partially routed net, in an implemented design.

This command can be used to define a routing path to assign to the FIXED_ROUTE property of a net, which can be saved to the XDC file for later reuse. Refer to *Vivado Design Suite Tutorial: Implementation (UG986)* for an example of manual routing and the use of the FIXED_ROUTE property.

You must define nodes for the start and end points of the routing path, and can specify the maximum and minimum number of nodes to use for the route path, including the start and end points. The nodes must be specified as objects returned by the `get_nodes` command. For unrouted net objects, since nodes have not been assigned to the net, the nodes can be found by association of the net to the bel_pin or site_pin:

- Net > Bel_Pin > Bel > Tile > Node

- Net > Site_Pin > Tile > Node

For partially routed nets, the nodes can be found associated directly to the net. Refer to the *Vivado Design Suite Properties Reference Guide (UG912)* for more information on the relationship between these objects.

The `find_routing_path` command returns a list of nodes representing the route path found from the start point to the end point, or returns "no path found" if the command runs but has no result, or returns an error if the command fails to run.

The returned list of nodes can be assigned to the FIXED_ROUTE property using the `set_property` command as shown in the example.

> 💡 **TIP:** *The* `report_property` *command does not return the string of the FIXED_ROUTE property. Use the* `get_property` *command instead.*

Arguments

`-allow_overlap` - (Optional) Enable a loose style of routing which can create conflicts that must be later resolved. These overlapping routes will need to be cleaned up to eliminate routing conflicts. Route conflicts can be identified using the `report_route_status` command.

`-max_nodes <arg>` - (Optional) Indicates the maximum number of nodes the route can contain, including the `-from` node and the `-to` node. The default is 100.

`-min_nodes <arg>` - (Optional) Indicates the minimum number of nodes the route can contain, including the `-from` node and the `-to` node. The default is 2, and the value specified must be >= 2.

> 💡 **TIP:** *This option can be used to generate a meandering route that will provide some added timing delay.*

`-from <arg>` - (Optional) The starting node of the route path. Nodes must be specified as objects returned by the `get_nodes` command.

`-to <arg>` - (Optional) The ending node of the route path. Nodes must be specified as objects returned by the `get_nodes` command.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example finds a routing path for the specified net, using one end as the `-from` point and the other end as the `-to` point, and assigns that path to the specified Tcl variable. Then it uses that Tcl variable to assign the path to the FIXED_ROUTE property of the net:

```
set fndPath [find_routing_path -from [lindex [get_nodes -of \
    [get_site_pins -of [get_nets wbOutputData_OBUF[14]]]] 0] -to \
    [lindex [get_nodes -of  [get_site_pins -of \
    [get_nets wbOutputData_OBUF[14]]]] 1]]
set_property FIXED_ROUTE $fndPath [get_nets wbOutputData_OBUF[14]]
```

See Also

- get_bel_pins
- get_nets
- get_nodes
- get_property
- get_site_pins
- report_property
- report_route_status
- set_property

Send Feedback

find_top

Find top module candidates in the supplied files, fileset, or active fileset. Returns a rank ordered list of candidates.

Syntax

```
find_top [-fileset <arg>] [-files <args>] [-return_file_paths] [-quiet]
    [-verbose]
```

Usage

Name	Description
`[-fileset]`	Fileset to parse to search for top candidates
`[-files]`	Files to parse to search for top candidates
`[-return_file_paths]`	For each top returned, also include the associated file path. The returned value will be a single list of strings, alternating top names and file paths.
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution

Categories

Project

Description

Find the most likely candidates for the top module in the files defined in the current fileset, or in the specified fileset, or in the specified list of files.

The command returns an ordered list of modules that the tool identifies as the best candidates for the top-level of the design. You can use the `lindex` command, and choose index 0 to select the best candidate for the top module.

Arguments

`-fileset <arg>` - (Optional) Search the specified simulation or source fileset for top module candidates. The default is to search the current fileset of the current design.

`-files <arg>` - (Optional) Find the top module candidates in the specified list of files.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the set_msg_config command.

Examples

The following example chooses the best top module of the current design for synthesis:

```
synth_design -top [lindex [find_top] 0]
```

Note: Since find_top returns multiple possible candidates, choosing index 0 chooses the best top candidate for synthesis.

The following example returns the best top module candidate from the specified list of files:

```
find_top -files [get_files -filter {NAME =~ *or1200*}]
```

The following example sets the results of find_top into the variable $topVar, then uses that variable to define the top module in the current fileset using the set_property command:

```
set topVar [ find_top -files [get_files -filter {NAME =~ *usbf*} ] ]
usbf_top
set_property top $topVar [current_fileset]
```

See Also

- set_property
- synth_design

flush_vcd

Flush VCD simulation output to the VCD output file (equivalent of $dumpflush verilog system task)

Syntax

```
flush_vcd [-quiet] [-verbose]
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Simulation

Description

Flush HDL signal information currently in memory into the specified Value Change Dump (VCD) file.

VCD is an ASCII file containing header information, variable definitions, and the value change details of a set of HDL signals. The VCD file can be used to view simulation results in a VCD viewer, or to estimate the power consumption of the design.

Note: You must run the `open_vcd` command to open a VCD file to write to before using the `flush_vcd` command.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Send Feedback

Examples

The following example flushes the VCD buffer into the current VCD file:

```
flush_vcd
```

See Also

- open_vcd

Send Feedback

generate_base_platform

Generate a base platform based on a given routed checkpoint

Syntax

```
generate_base_platform [-source <arg>] [-reconfig_platform <arg>]
    [-base_platform <arg>] [-reconfig_platform_prefix <arg>] [-quiet]
    [-verbose]
```

Usage

Name	Description
[-source]	(Required) Specify routed checkpoint path
[-reconfig_platform]	(Required) Specify reconfigurable platform module name
[-base_platform]	(Optional) Specify the output file name, the default file name is 'base_platform'
[-reconfig_platform_prefix]	(Optional) Specify wrapper port name prefix from reconfigurable platform module, the default prefix is 'RL_'
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

FileIO

generate_hier_access

(User-written application) Generate sources for hierarchical access simulation

Syntax

```
generate_hier_access [-bypass <arg>] [-driver <arg>] [-directory <arg>]
     [-pseudo_top <arg>] [-testbench <arg>] [-log <arg>] [-quiet] [-verbose]
```

Returns

None

Usage

Name	Description
[-bypass]	Hierarchical access module name Default: xil_dut_bypass
[-driver]	Signal driver template module name Default: xil_bypass_driver
[-directory]	Output directory for the generated sources Default: current working directory
[-pseudo_top]	Top-level pseudo testbench module name Default: None
[-testbench]	User design testbench module name Default: None
[-log]	Simulator log containing hierarchical path information (required for the non-Vivado standalone flow only) Default: None
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

xilinxtclstore

Description

Generates sources for hierarchical access simulation.

Arguments

`[-bypass <arg> = xil_dut_bypass]` - (Optional) Hierarchical access module name.

`[-driver <arg> = xil_bypass_driver]` - (Optional) Signal driver template module name.

`[-directory <arg> = current working directory]` - (Optional) Output directory for the generated sources.

`[-pseudo_top <arg>]` - (Optional) Top-level pseudo testbench module name.

`[-testbench <arg>]` - (Optional) User design testbench module name.

`[-log <arg>]` - (Optional) Simulator log containing hierarchical path information (required for the non-Vivado standalone flow only).

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

generate_mem_files

Write all the simulation .mem files.

Syntax

```
generate_mem_files [-force] [-quiet] [-verbose] <directory>
```

Returns

The name of the directory.

Usage

Name	Description
[-force]	Overwrite existing .mem files
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<directory>	Directory for exporting .mem files. Values: A directory with alphanumeric characters.

Categories

FileIO, Simulation

Description

For embedded processor based designs, with associated Executable Linkable Files (ELF) from the Software Development Kit (SDK), this command merges the Block Memory Map (BMM) for the design with the program data in the ELF file to generate memory (MEM) files for use during simulation.

The MEM file is a text file that describes how individual Block RAMs on the Xilinx device are grouped together to form a contiguous address space called an Address Block, with the ELF data mapped into the memory.

The file names and the number of MEM files generated is determined by the memory map data specified by the processor system IP cores, or IP integrator block designs.

This command returns the directory where the MEM files are written, or returns an error if it fails.

Arguments

-force - (Optional) Overwrite the specified output directory if it already exists.

Send Feedback

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<directory> - (Required) The name of the directory to write the memory files into.

Example

The following example merges the block RAM map with the ELF file data and generates MEM files in the specified directory for use during simulation:

```
generate_mem_files C:/Data/gen_mem
```

See Also

- write_bmm
- write_mem_info

Send Feedback

generate_pblock

Generate PBLOCK by exclude static

Syntax

```
generate_pblock [-cell <arg>] [-inverse_pblock <arg>]
    [-nested_pblock <arg>] [-nested_width <arg>] [-quiet] [-verbose]
```

Usage

Name	Description
[-cell]	Specify cell to add to pblock.
[-inverse_pblock]	Specify name of the inverse pblock. The pblock will cover everything that the static pblock does not.
[-nested_pblock]	Specify name of nested pblock inside inverse_pblock left adjacent to static pblock.
[-nested_width]	Specify the width of the nested pblock. The nested pblock height is the same as the adjacent static pblock. Default: 3
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

FileIO

Send Feedback

generate_peripheral

Generate output products for peripheral object.

Syntax

```
generate_peripheral [-driver] [-example_design] [-bfm_example_design]
    [-debug_hw_example_design] [-enable_interrupt] [-force] [-quiet]
    [-verbose] <peripheral>
```

Usage

Name	Description
[-driver]	Generate driver for peripheral.
[-example_design]	Generate all supported example designs for peripheral.
[-bfm_example_design]	Generate bfm simulation example design for peripheral.
[-debug_hw_example_design]	Generate debug hardware example design for peripheral.
[-enable_interrupt]	Generate peripheral with interrupt suppport.
[-force]	Overwrite the existing IP in the repository.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<peripheral>	peripheral object

Categories

Project, IPFlow, CreatePeripheral

Description

Generate the output products for the specified peripheral object. The output products are written to the IP repository location specified when the IP is created by the `create_peripheral` command, under the name of the IP as specified at creation.

Arguments

`-driver` - (Optional) Create software driver files containing offsets of software addressable registers in the generated peripheral, as well as masks and register access macros or utility functions. The software driver self test example file contains self test example code to test various hardware features of the peripheral.

`-example_design` - (Optional) Write example design data for the specified peripheral. This includes the `design.tcl` to create a block design incorporating the new peripheral in IP integrator, and a test bench, called `design_tb.v`, for simulating the example design.

Send Feedback

Note: This option is not currently supported for AXI peripherals that implement the AXI stream interface as defined by the `-axi_type` option of the `add_peripheral_interface` command.

`-bfm_example_design` - (Optional) Create a Tcl script to generate a block design using the IP integrator of the Vivado Design Suite ad a bus functional model (BFM) test bench to test the read and write operations of the AXI peripheral.

Note: AXI4 BFM requires a license for use during simulation.

`-debug_hw_example_design` - (Optional) Create a Tcl script to generate a block design for debugging the AXI peripheral using the JTAG-to-AXI debug core in the Hardware Manager feature of the tool. Refer to the *Vivado Design Suite User Guide: Programming and Debugging (UG908)* for more information on working with the Hardware Manager.

`-enable_interrupt` - (Optional) Add an interrupt pin and supporting logic, to enable the interrupt operation on the peripheral.

`-force` - (Optional) Overwrite any existing output products in the IP repository even if they are current.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<peripheral> - (Required) The peripheral object to generate output products for. The peripheral is created with the `create_peripheral` command, and should be captured in a Tcl variable at the time it is created to facilitate further processing by this and other related commands. See the example below.

Example

This example creates a new AXI peripheral, with the VLNV attribute as specified, and captures the peripheral object in a Tcl variable for later processing, then adds AXI slave interfaces to the peripheral, and generates the output products for the peripheral:

```
set perifObj [ create_peripheral {myCompany.com} {user} {testAXI1} \
    {1.3} -dir {C:/Data/new_periph} ]
add_peripheral_interface {S0_AXI} -interface_mode {slave} \
    -axi_type {lite} $perifObj
add_peripheral_interface {S1_AXI} -interface_mode {slave} \
    -axi_type {lite} $perifObj
```

Send Feedback

```
generate_peripheral -driver -bfm_example_design \
    -enable_interrupt $perifObj
write_peripheral $perifObj
set_property ip_repo_paths  C:/Data/new_periph [current_fileset]
update_ip_catalog -rebuild
```

See Also

- add_peripheral_interface
- create_peripheral
- write_peripheral

generate_reports

Generate a set of configurable report objects

Syntax

```
generate_reports [-jobs <arg>] [-quiet] [-verbose] <report_configs>...
```

Usage

Name	Description
`[-jobs]`	Number of jobs Default: 1
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<report_configs>`	List of configurable report objects to generate

Categories

[Object](#), [Report](#)

Description

Generates specified report objects as created by the `create_report_config` command.

Because the report objects are associated with specific steps of synthesis or implementation runs, those steps must be completed prior to the generation of the report. If a step is not completed, or the report is not enabled, then the `generate_report` command will return an error.

Arguments

`-jobs <arg>` - (Optional) Specifies the number of jobs to use to generate the specified report objects.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<report_configs> - (Required) Specifies the list of report objects to generate. The reports must be specified as objects returned by the `get_report_configs` command.

Examples

The following example generates the specified report object:

```
generate_report [get_report_configs post_route_datasheet]
```

> **TIP:** *If the report is already GENERATED according to the STATE property, the report will not be regenerated.*

See Also

- create_report_config
- delete_report_configs
- get_report_configs

generate_rl_platform

Generate new platform based on base platform and wrapper module

Syntax

```
generate_rl_platform [-use_source <arg>] [-reconfig_platform <arg>]
    [-base_platform <arg>] [-platform <arg>] [-quiet] [-verbose]
```

Usage

Name	Description
[-use_source]	Specify wrapper checkpoint path
[-reconfig_platform]	(Required) Specify reconfigurable platform module name
[-base_platform]	(Required) Specify base platform checkpoint path
[-platform]	(Optional) Specify new platform checkpoint path, the default file name is 'rl_platform.dcp'
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

FileIO

Send Feedback

generate_shx_platform

Move HD.RECONFIGURABLE and related properties to sub-cells

Syntax

```
generate_shx_platform [-base_platform <arg>] [-wrapper <arg>]
    [-output <arg>] [-reconfig_platform <arg>] [-quiet] [-verbose]
```

Usage

Name	Description
[-base_platform]	Specify Base Platform DCP path.
[-wrapper]	Specify wrapper DCP path
[-output]	Specify output DCP name.
[-reconfig_platform]	Specify reconfigurable platform name
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

FileIO

generate_target

Generate target data for the specified source

Syntax

```
generate_target [-force] [-quiet] [-verbose] <name> <objects>
```

Usage

Name	Description
[-force]	Force target data regeneration
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<name>	List of targets to be generated, or 'all' to generate all supported targets
<objects>	The objects for which data needs to be generated

Categories

Project, IPFlow, IPIntegrator

Description

This command generates target data for the specified IP objects (`get_ips`) or source file for IP cores (`.xci` and `.xco`), DSP modules (`.slx` or `.mdl`), or block designs (`.bd`). The target data that is generated are the files necessary to support the IP or block design through the FPGA design flow.

The instantiation template, synthesis netlist, and simulation netlist are standard targets. However, each IP in the catalog may also support its own set of targets. You can view the available targets on an object by examining the SUPPORTED_TARGETS property, or you can use the `list_targets` command to list the targets for design source file.

Arguments

`-force` - (Optional) Force target data regeneration, and overwrite any existing target data files. Without `-force`, the tool will not regenerate any target data that is up-to-date.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - (Required) The names of the types of target data to create for the specified source. The specific targets supported by an IP core are listed in the SUPPORTED_TARGETS property on the object. You can query this property to see which targets a specific object supports. Standard values are:

- `all` - Generate all targets for the specified IP or file, except the example project.

  Note: You can generate an example project for an IP core using the `open_example_project` command.

- `instantiation_template` - Generate the Instantiation template used to add the RTL module definition for the IP core into the current design. The instantiation template can be copied into any desired level of the design hierarchy.

- `synthesis` - Synthesis targets deliver HDL files that are used during synthesis for native IP, or deliver a synthesized netlist file (DCP) generated by Vivado synthesis.

- `simulation` - Simulation targets deliver HDL files that are used in simulation.

- `implementation` - Implementation generates the necessary data for implementing the IP core, DSP module, or Embedded Processor in the current design.

`<objects>` - (Required) The objects to generate the target from. Supported objects can include IP core objects (`get_ips`), or the source files (`.xci` or `.xco`), block design files (`.bd`) from IP integrator, and DSP modules (`.slx` or `.mdl`) imported from System Generator.

Note: Use `get_files` to specify a file object, rather than specifying a file name.

Examples

This example generates the change log for all of the IP cores in the current project, forcing regeneration of any targets that are up-to-date:

```
generate_target changelog [get_ips] -force
```

The following example generates the instantiation template and synthesis targets for all of the IP cores in the current project:

```
generate_target {instantiation_template synthesis} [get_ips]
```

> **TIP:** *Note the use of the braces to pass the list of targets to the command. The absence of the `-force` option means that only out-of-date targets will be regenerated.*

The following example generates all targets for the specified block design:

```
generate_target all \
[get_files  C:/Data/project_mb/project_mb.srcs/sources_1/bd/base_mb/
base_mb.bd]
```

IMPORTANT! *The use of* `get_ips` *is not supported to generate targets for individual IP within block designs. The tool will return an error.*

The following queries the SUPPORTED_TARGETS property of the specified IP object, and then generates the example project for the IP:

```
get_property SUPPORTED_TARGETS [get_ips blk_mem*]
open_example_project -dir C:/Data/examples -force [get_ips blk_mem*]
```

See Also

- add_files
- create_ip
- create_sysgen
- import_ip
- list_targets
- open_example_project
- read_ip
- report_property
- reset_target

Send Feedback

get_bd_addr_segs

Get a list of segments

Syntax

```
get_bd_addr_segs [-regexp] [-hierarchical] [-filter <arg>]
    [-of_objects <args>] [-excluded] [-addressed] [-unaddressed]
    [-addressing] [-addressables] [-quiet] [-verbose] [<patterns>]
```

Returns

List of segment objects, "" if failed.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-hierarchical]	Hierarchical cells included
[-filter]	Filter list with expression
[-of_objects]	Get segments of these segments, interfaces, or registers
[-excluded]	Get excluded mapped segments -of_objects
[-addressed]	Get addressed segments of given -of_objects
[-unaddressed]	Get unaddressed segments of given objects
[-addressing]	Get addressing segments of given -of_objects
[-addressables]	Get addressable segments of given -of_objects
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match engine names against patterns Default: *

Categories

[IPIntegrator](#)

Description

Get a list of address segments in the current IP integrator subsystem design.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Send Feedback

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-hierarchical` - (Optional) Get address segments from all levels of the IP integrator subsystem design hierarchy, or current instance. Without this argument, the command will only get address segments from the top of the subsystem design hierarchy. When using `-hierarchical`, the search pattern should not contain a hierarchy separator because the search pattern is applied at each level of the hierarchy, not to the full hierarchical name. For instance, searching for U1/* searches each level of the hierarchy for objects with U1/ in the name. This may not return the intended results.

Note: When used with `-regexp`, the specified search string is matched against the full hierarchical name, and the U1/* search pattern will work as intended.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_bd_addr_segs` based on property values on the address segments. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the IP integrator address segments object, "OFFSET", "RANGE" and "USAGE" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Send Feedback

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <arg>` - (Optional) Get the address segments that are assigned to the specified address spaces, as returned by the `get_bd_addr_spaces` command, or of cells or interface pins, as returned by `get_bd_cells` and `get_bd_intf_pins`. You can also get slave address_segments of associated master address segments using `get_bd_addr_segs`.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-excluded` - (Optional) Get the address segments that are excluded from the address spaces specified by the `-of_objects` option.

`-addressing` - (Optional) Get addressing segments of the address spaces specified by the `-of_objects` option. This reports all available segments of the address spaces.

`-addressed` - (Optional) Get addressed segments of the address spaces specified by the `-of_objects` option. This reports the segments of the address spaces that are used.

`-addressables` - (Optional) Get addressable segments, that are not already occupied by address segments, of the address spaces of the objects specified by the `-of_objects` option.

Note: `-addressables`, `-addressed`, and `-addressing` are mutually exclusive, and cannot be used together.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match address segments against the specified patterns. The default pattern is the wildcard '*' which gets a list of all address segments in the current IP subsystem design. More than one pattern can be specified to find multiple address segments based on different search criteria.

Note: You must enclose multiple search patterns in braces {} to present the list as a single element.

Examples

The following example gets the address segments of the specified address spaces:

```
get_bd_addr_segs -of_objects [get_bd_addr_spaces -of_objects \
[get_bd_cells /microblaze_1]]
/microblaze_1/Data/SEG1
/microblaze_1/Data/SEG3
/microblaze_1/Instruction/SEG2
```

Note: If there are no objects matching the pattern you will get a warning.

See Also

- assign_bd_address
- create_bd_addr_seg
- exclude_bd_addr_seg
- get_bd_addr_spaces
- get_bd_cells
- get_bd_intf_pins
- include_bd_addr_seg

Send Feedback

get_bd_addr_spaces

Get a list of addr_spaces

Syntax

```
get_bd_addr_spaces [-regexp] [-hierarchical] [-filter <arg>]
    [-of_objects <args>] [-quiet] [-verbose] [<patterns>]
```

Returns

List of addr_space objects, "" if failed.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-hierarchical]	Hierarchical cells included
[-filter]	Filter list with expression
[-of_objects]	Get addr_spaces of these segments or interfaces
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match engine names against patterns Default: *

Categories

[IPIntegrator](#)

Description

Get a list of address spaces in the current IP integrator subsystem design.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-hierarchical` - (Optional) Get address spaces from all levels of the IP integrator subsystem design hierarchy, or current instance. Without this argument, the command will only get address spaces from the top of the design hierarchy. When using `-hierarchical`, the search pattern should not contain a hierarchy separator because the search pattern is applied at each level of the hierarchy, not to the full hierarchical name. For instance, searching for U1/* searches each level of the hierarchy for objects with U1/ in the name. This may not return the intended results.

Note: When used with `-regexp`, the specified search string is matched against the full hierarchical name, and the U1/* search pattern will work as intended.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_bd_addr_spaces` based on property values on the objects. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the IP integrator address space object, "NAME", "RANGE" and "OFFSET" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <arg> - (Optional) Get the address spaces of the specified IP integrator address segment, cell, or interface pin objects.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match address spaces against the specified patterns. The default pattern is the wildcard '*' which gets a list of all address spaces in the current IP subsystem design. More than one pattern can be specified to find multiple address spaces based on different search criteria.

Note: You must enclose multiple search patterns in braces {} to present the list as a single element.

Examples

The following example lists all of the address spaces in the current IP integrator subsystem design, listing one per line:

```
join [get_bd_addr_spaces] \n
/mdm_1/S_AXI
/microblaze_1/Data
/microblaze_1/Instruction
/microblaze_1_axi_intc/s_axi
/microblaze_1_local_memory/dlmb_bram_if_cntlr/SLMB
/microblaze_1_local_memory/dlmb_bram_if_cntlr/SLMB1
/microblaze_1_local_memory/dlmb_bram_if_cntlr/SLMB2
/microblaze_1_local_memory/dlmb_bram_if_cntlr/SLMB3
/microblaze_1_local_memory/dlmb_bram_if_cntlr/S_AXI_CTRL
/microblaze_1_local_memory/ilmb_bram_if_cntlr/SLMB
/microblaze_1_local_memory/ilmb_bram_if_cntlr/SLMB1
/microblaze_1_local_memory/ilmb_bram_if_cntlr/SLMB2
/microblaze_1_local_memory/ilmb_bram_if_cntlr/SLMB3
/microblaze_1_local_memory/ilmb_bram_if_cntlr/S_AXI_CTRL
/microblaze_1_local_memory/lmb_bram/S_1
```

Note: If there are no objects matching the pattern you will get a warning.

The following example returns all of the properties attached to the third in a list, or index 2, of all address spaces in the current subsystem design:

```
report_property -all [lindex [get_bd_addr_spaces] 2 ]
Property    Type      Read-only   Visible    Value
CLASS       string    true        true       bd_addr_space
NAME        string    false       true       /microblaze_1/Instruction
OFFSET      string    false       true
PATH        string    true        true       /microblaze_1/Instruction
RANGE       string    false       true       -1
TYPE        string    false       true
```

Send Feedback

See Also

- assign_bd_address
- create_bd_addr_seg
- exclude_bd_addr_seg
- get_bd_addr_segs
- get_bd_cells
- get_bd_intf_pins
- include_bd_addr_seg

get_bd_cells

Get a list of block diagram cells

Syntax

```
get_bd_cells [-regexp] [-hierarchical] [-filter <arg>] [-of_objects <args>]
     [-quiet] [-verbose] [<patterns>]
```

Returns

List of block diagram cell objects, "" if failed.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-hierarchical]	Hierarchical cells included
[-filter]	Filter list with expression
[-of_objects]	Get cells of these pins or nets
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match engine names against patterns Default: *

Categories

IPIntegrator

Description

Get a list of cells in the current IP integrator subsystem design, or current subsystem instance. IP integrator subsystem cells are either IP cores drawn from the IP integrator catalog, or hierarchical modules created in the subsystem design with the `create_bd_cell` command.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Send Feedback

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-hierarchical` - (Optional) Get cells from all levels of the IP integrator subsystem design hierarchy, or current instance. Without this argument, the command will only get cells from the top of the design hierarchy. When using `-hierarchical`, the search pattern should not contain a hierarchy separator because the search pattern is applied at each level of the hierarchy, not to the full hierarchical cell name. For instance, searching for U1/* searches each level of the hierarchy for instances with U1/ in the name. This may not return the intended results.

Note: When used with `-regexp`, the specified search string is matched against the full hierarchical name, and the U1/* search pattern will work as intended.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_bd_cells` based on property values on the block design cells. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the IP integrator cell object, "VLNV", "TYPE" and "LOCATION" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <arg>` - (Optional) Get the subsystem cells that are connected to the specified pin or net objects, as returned by the `get_bd_nets` and `get_bd_pins`, or `get_bd_intf_pins` commands.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match subsystem cells against the specified patterns. The default pattern is the wildcard '*' which gets a list of all cells in the current IP subsystem design. More than one pattern can be specified to find multiple cells based on different search criteria.

Note: You must enclose multiple search patterns in braces {} to present the list as a single element.

Examples

The following example gets a list of cells that include the specified IP integrator subsystem pin, and sorts the list to remove duplicate entries:

```
lsort -unique [get_bd_cells -of_objects [get_bd_pins -hierarchical *aclk*]]
```

Note: If there are no cells matching the pattern you will get a warning.

The following example gets a list of all cells in all levels of the subsystem design hierarchy, and then filters the list to include only those cells whose name includes the specified text, or hierarchy:

```
get_bd_cells -hierarchical -filter {NAME=~"/newMod1/*"}
```

See Also

- create_bd_cell
- get_bd_intf_pins

- get_bd_nets
- get_bd_pins
- list_property
- report_property

get_bd_designs

Get a list of designs

Syntax

```
get_bd_designs [-regexp] [-filter <arg>] [-of_objects <args>] [-quiet]
    [-verbose] [<patterns>...]
```

Returns

List of design objects, "" if failed.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-filter]	Filter list with expression
[-of_objects]	Get diagrams of these bd-cells or pins or nets
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match engine names against patterns Default: *

Categories

[IPIntegrator](#)

Description

Gets a list of IP subsystem designs open in the current project that match a specified search pattern. The default command gets a list of all open IP subsystem designs in the project.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_bd_designs` based on property values on the block designs. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the "IP subsystem design" object, "NAME", and "FILE_NAME" are two of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <args> - (Optional) Get the BD designs associated with the specified bd_cell, bd_pins, or bd_net objects.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match designs against the specified patterns. The default pattern is the wildcard '*' which gets all IP subsystem designs. More than one pattern can be specified to find multiple designs based on different search criteria.

Examples

The following example gets all open IP subsystem designs in the current project:

```
get_bd_designs
```

See Also

- create_bd_design
- current_bd_design
- open_bd_design
- report_property

Send Feedback

get_bd_intf_nets

Get a list of intf_nets

Syntax

```
get_bd_intf_nets [-regexp] [-hierarchical] [-filter <arg>]
    [-boundary_type <arg>] [-of_objects <args>] [-quiet] [-verbose]
    [<patterns>]
```

Returns

List of pin objects, "" if failed.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-hierarchical]	Hierarchical cells included
[-filter]	Filter list with expression
[-boundary_type]	Used when source object is on a hierarchical block's interface pin. Valid values are 'upper', 'lower', or 'both'. If 'lower' boundary, searches from the lower level of hierarchy onwards. This option is only valid for connected_to relations.
[-of_objects]	One or a list of cells or interface pins/ports objects. List must be of one object type.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match engine names against patterns Default: *

Categories

IPIntegrator

Description

Gets a list of interface nets in the current IP integrator subsystem design that match a specified search pattern. The default command gets a list of all interface nets in the subsystem design.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Send Feedback

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-hierarchical` - (Optional) Get interface nets from all levels of the IP integrator subsystem design hierarchy. Without this argument, the command will only get interface nets from the top of the subsystem design, or from the current subsystem instance. When using `-hierarchical`, the search pattern should not contain a hierarchy separator because the search pattern is applied at each level of the hierarchy, not to the full hierarchical name. For instance, searching for /bridge_1/* searches each level of the hierarchy for nets with /bridge_1/ in the name. This may not return the intended results.

Note: When used with `-regexp`, the specified search string is matched against the full hierarchical name, and the U1/* search pattern will work as intended.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_bd_intf_nets` based on property values on the block design interface nets. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the IP integrator interface nets object, "NAME" and "PATH" are two of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Send Feedback

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-boundary_type [ upper | lower | both ]` - (Optional) Gets the net at the level (upper) of a specified hierarchical pin, at the level below (lower) the pin or port, or both the level of and the level below. Valid values are upper, lower, or both. The default value is upper.

Note: This argument must be used with the `-of_objects` argument to specify the hierarchical pin.

`-of_objects` <args> - (Optional) Get a list of the nets connected to the specified IP integrator subsystem cell, pin, or port objects.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match IP subsystem interface nets against the specified patterns. The default pattern is the wildcard '*' which returns a list of all interface nets in the current IP integrator subsystem design. More than one pattern can be specified to find multiple nets based on different search criteria.

Note: You must enclose multiple search patterns in braces {} to present the list as a single element.

Examples

The following example gets the interface net attached to the specified pin of an IP integrator hierarchical module, and returns both the net at the level of the hierarchical module, and the net inside the hierarchical module:

```
get_bd_intf_nets -boundary_type both -of_objects [get_bd_pins /newMod1/aclk]
```

Note: If there are no interface nets matching the pattern you will get a warning.

Send Feedback

The following example returns a list of all interface nets at all levels of the IP integrator subsystem design hierarchy:

```
get_bd_intf_nets -hierarchical
```

See Also

- connect_debug_port
- get_cells
- get_clocks
- get_pins
- get_ports
- list_property
- report_property

get_bd_intf_pins

Get a list of intf_pins

Syntax

```
get_bd_intf_pins [-regexp] [-hierarchical] [-filter <arg>]
    [-of_objects <args>] [-quiet] [-verbose] [<patterns>]
```

Returns

List of pin objects, "" if failed.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-hierarchical]	Hierarchical cells included
[-filter]	Filter list with expression
[-of_objects]	One or a list of cells, interface nets or pins objects. List must be of one object type.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match engine names against patterns Default: *

Categories

IPIntegrator

Description

Gets a list of interface pin objects on the current IP subsystem design that match a specified search pattern. The default command gets a list of all interface pins in the subsystem design.

The external connections of an IP integrator cell, or hierarchical module, are pins and interface pins. The external connections in an IP subsystem design are ports, or interface ports. Use the `get_bd_ports` and `get_bd_intf_ports` commands to select the port objects.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-hierarchical` - (Optional) Get interface pins from all levels of the IP integrator subsystem design hierarchy. Without this argument, the command will only get interface pins from the top of the subsystem design, or from the current subsystem instance. When using `-hierarchical`, the search pattern should not contain a hierarchy separator because the search pattern is applied at each level of the hierarchy, not to the full hierarchical name. For instance, searching for /bridge_1/* searches each level of the hierarchy for interface pins with /bridge_1/ in the name. This may not return the intended results.

Note: When used with `-regexpr`, the specified search string is matched against the full hierarchical name, and the U1/* search pattern will work as intended.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_bd_intf_pins` based on property values on the block design interface pins. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the IP integrator interface pins object, "DIR" and "TYPE" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <arg>` - (Optional) Get the pins connected to the specified bd_pin, bd_cell, or bd_intf_net.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match IP integrator subsystem interface pins against the specified patterns. The default pattern is the wildcard '*' which gets a list of all interface pins in the current IP integrator subsystem design, or instance. More than one pattern can be specified to find multiple pins based on different search criteria.

Note: You must enclose multiple search patterns in braces {} to present the list as a single element.

Examples

The following example gets a list of interface pins attached to the specified cell:

```
get_bd_intf_pins -of [get_bd_cells new_vidOut_1]
```

Note: If there are no interface pins matching the pattern, the tool will return a warning.

The following example gets a list of all interface pins, throughout the hierarchy of the IP integrator subsystem design, which match the specified name pattern:

```
get_bd_intf_pins -hierarchical m_apb*
```

The following example gets a list of interface pins attached to the specified subsystem net:

```
get_bd_intf_pins -of [get_bd_intf_nets vidout_1_vtg_ce]
```

See Also

- create_bd_net

- create_bd_pin
- create_bd_port
- get_bd_intf_ports
- get_bd_nets
- get_bd_pins
- list_property
- report_property

Send Feedback

get_bd_intf_ports

Get a list of intf_ports

Syntax

```
get_bd_intf_ports [-regexp] [-filter <arg>] [-of_objects <args>] [-quiet]
    [-verbose] [<patterns>]
```

Returns

List of port objects, "" if failed.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-filter]	Filter list with expression
[-of_objects]	One or a list of interface nets or ports objects.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match engine names against patterns Default: *

Categories

IPIntegrator

Description

Gets a list of interface port objects in the current IP integrator subsystem design that match a specified search pattern. The default command gets a list of all interface ports in the subsystem design.

The external connections in an IP subsystem design are ports, or interface ports. The external connections in an IP integrator cell, or hierarchical module, are pins and interface pins. Use the `get_bd_pins` and `get_bd_intf_pins` commands to select the pin objects.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_bd_intf_ports` based on property values on the block design interface ports. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the IP subsystem interface port object, "DIR", "MODE", and "LOCATION" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <arg> - (Optional) Get the interface ports connected to the specified IP subsystem interface nets returned by `get_bd_intf_nets`, or of the ports that are part of an interface port, as returned by the `get_bd_ports` command.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match interface ports against the specified patterns. The default pattern is the wildcard '*' which gets a list of all interface ports in the subsystem design. More than one pattern can be specified to find multiple interface ports based on different search criteria.

Note: You must enclose multiple search patterns in braces {} to present the list as a single element.

Examples

The following example gets the interface ports in the subsystem design that operate in Master mode:

```
get_bd_intf_ports -filter {MODE=="master"}
```

Note: If there are no interface ports matching the pattern, the tool will return a warning.

This example returns the block design interface port that the specified bd_port is part of:

```
get_bd_intf_ports -of [get_bd_ports sys_diff_clock_clk_n]
```

See Also

- create_bd_intf_net
- create_bd_intf_pin
- create_bd_intf_port
- get_bd_intf_nets
- get_bd_intf_pins
- get_bd_nets
- get_bd_pins
- get_bd_ports
- list_property
- report_property

Send Feedback

get_bd_nets

Get a list of nets

Syntax

```
get_bd_nets [-regexp] [-hierarchical] [-filter <arg>]
    [-boundary_type <arg>] [-of_objects <args>] [-quiet] [-verbose]
    [<patterns>]
```

Returns

List of pin objects, "" if failed.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-hierarchical]	Hierarchical cells included
[-filter]	Filter list with expression
[-boundary_type]	Used when source object is on a hierarchical block's pin. Valid values are 'upper', 'lower', or 'both'. If 'lower' boundary, searches from the lower level of hierarchy onwards. This option is only valid for connected_to relations.
[-of_objects]	One or a list of cells or pins/ports objects. List must be of one object type.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match engine names against patterns Default: *

Categories

IPIntegrator

Description

Gets a list of nets in the current IP integrator subsystem design that match a specified search pattern. The default command gets a list of all nets in the subsystem design.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Send Feedback

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-hierarchical` - (Optional) Get nets from all levels of the IP integrator subsystem design hierarchy. Without this argument, the command will only get nets from the top of the subsystem design, or from the current subsystem instance. When using `-hierarchical`, the search pattern should not contain a hierarchy separator because the search pattern is applied at each level of the hierarchy, not to the full hierarchical name. For instance, searching for /bridge_1/* searches each level of the hierarchy for nets with /bridge_1/ in the name. This may not return the intended results.

Note: When used with `-regexp`, the specified search string is matched against the full hierarchical name, and the U1/* search pattern will work as intended.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_bd_nets` based on property values on the block design nets. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the IP integrator subsystem nets object, "NAME" and "PATH" are two of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Send Feedback

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-boundary_type [ upper | lower | both ]` - (Optional) Gets the net at the level (upper) of a specified hierarchical pin, at the level below (lower) the pin or port, or both the level of and the level below. Valid values are upper, lower, or both. The default value is upper.

Note: This argument must be used with the `-of_objects` argument to specify the hierarchical pin.

`-of_objects` <args> - (Optional) Get a list of the nets connected to the specified cell, pin, or port objects.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match IP subsystem nets against the specified patterns. The default pattern is the wildcard '*' which returns a list of all nets in the current IP integrator subsystem design. More than one pattern can be specified to find multiple nets based on different search criteria.

Note: You must enclose multiple search patterns in braces {} to present the list as a single element.

Examples

The following example gets the net attached to the specified pin of an IP integrator hierarchical module, and returns both the net at the level of the hierarchical module, and the net inside the hierarchical module:

```
get_bd_nets -boundary_type both -of_objects [get_bd_pins /newMod1/aclk]
```

Note: If there are no nets matching the pattern you will get a warning.

The following example returns a list of all nets at all levels of the IP integrator subsystem design hierarchy:

```
get_bd_nets -hierarchical
```

See Also

- connect_debug_port
- get_cells
- get_clocks
- get_pins
- get_ports
- list_property
- report_property

Send Feedback

get_bd_pins

Get a list of pins

Syntax

```
get_bd_pins [-regexp] [-hierarchical] [-filter <arg>] [-of_objects <args>]
    [-quiet] [-verbose] [<patterns>]
```

Returns

List of pin objects, "" if failed.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-hierarchical]	Hierarchical cells included
[-filter]	Filter list with expression
[-of_objects]	One or a list of cells, nets or interface pins objects. List must be of one object type.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match engine names against patterns Default: *

Categories

[IPIntegrator](#)

Description

Gets a list of pin objects on the current IP subsystem design that match a specified search pattern. The default command gets a list of all pins in the subsystem design.

The external connections in an IP integrator cell, or hierarchical module, are pins and interface pins. The external connections in an IP subsystem design are ports, or interface ports. Use the `get_bd_ports` and `get_bd_intf_ports` commands to select the port objects.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-hierarchical` - (Optional) Get pins from all levels of the IP integrator subsystem design hierarchy. Without this argument, the command will only get pins from the top of the subsystem design, or from the current subsystem instance. When using `-hierarchical`, the search pattern should not contain a hierarchy separator because the search pattern is applied at each level of the hierarchy, not to the full hierarchical name. For instance, searching for /bridge_1/* searches each level of the hierarchy for pins with /bridge_1/ in the name. This may not return the intended results.

Note: When used with `-regexpr`, the specified search string is matched against the full hierarchical name, and the U1/* search pattern will work as intended.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by get_bd_pins based on property values on the block design pins. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the block design pins object, "DIR" and "TYPE" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <arg>` - (Optional) Get the pins connected to the specified IP subsystem interface pins, bd_intf_pin, bd_ cell, or bd_net.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match IP integrator subsystem pins against the specified patterns. You must specify the search pattern using the name convention of <bd_cell_name> /<bd_pin_name>. In this case, using the wildcard '*/*' would return all of the bd_pin objects on all of the bd_cells in the diagram.

Note: More than one pattern can be specified to find multiple pins based on different search criteria. You must enclose multiple search patterns in braces {} to present the list as a single element.

Examples

The following example gets a list of pins attached to the specified cell:

```
get_bd_pins -of [get_bd_cells new_vidOut_1]
```

Note: If there are no pins matching the pattern, the tool will return a warning.

The following example gets a list of all pins, throughout the hierarchy of the IP integrator subsystem design, which match the specified name pattern:

```
get_bd_pins -hierarchical LMB*
```

The following example gets a list of pins attached to the specified subsystem net:

```
get_bd_pins -of [get_bd_nets vidout_1_vtg_ce]
```

See Also

- create_bd_net
- create_bd_pin
- create_bd_port
- get_bd_intf_pins
- get_bd_intf_ports
- get_bd_nets
- list_property
- report_property

get_bd_ports

Get a list of ports

Syntax

```
get_bd_ports [-regexp] [-filter <arg>] [-of_objects <args>] [-quiet]
    [-verbose] [<patterns>]
```

Returns

List of port objects, "" if failed.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-filter]	Filter list with expression
[-of_objects]	One or a list of nets or interface ports objects.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match engine names against patterns Default: *

Categories

IPIntegrator

Description

Gets a list of port objects in the current IP integrator subsystem design that match a specified search pattern. The default command gets a list of all ports in the subsystem design.

The external connections in an IP subsystem design are ports, or interface ports. The external connections in an IP integrator cell, or hierarchical module, are pins and interface pins. Use the `get_bd_pins` and `get_bd_intf_pins` commands to select the pin objects.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Send Feedback

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_bd_ports` based on property values on the block design ports. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the IP subsystem port object, "DIR", "MODE", and "LOCATION" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <arg> - (Optional) Get the ports connected to the specified IP subsystem nets returned by `get_bd_nets`, or ports that are part of an interface port as returned by `get_bd_intf_ports`.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match ports against the specified patterns. The default pattern is the wildcard '*' which gets a list of all ports in the subsystem design. More than one pattern can be specified to find multiple ports based on different search criteria.

Note: You must enclose multiple search patterns in braces {} to present the list as a single element.

Examples

The following example gets the ports connected to the specified IP subsystem net:

```
get_bd_ports -of_objects [get_bd_nets bridge_1_apb_m]
```

Note: If there are no ports matching the pattern, the tool will return a warning.

See Also

- create_bd_net
- create_bd_pin
- create_bd_port
- get_bd_intf_pins
- get_bd_intf_ports
- get_bd_nets
- get_bd_pins
- list_property
- report_property

get_bd_regs

Get a list of registers

Syntax

```
get_bd_regs [-of_objects <args>] [-quiet] [-verbose]
```

Returns

List of register objects, "" if failed.

Usage

Name	Description
[-of_objects]	Get registers of segments, interface pins, external interface ports
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

IPIntegrator

Description

Get a list of registers using the objects of address segments, interface pins, and external interface ports. The register objects will provide the register's display name and the memory offset address. There is no default behavior for this command as it requires the use of `-of_objects` argument with a valid value that is an object of the address segments, interface pins, or external interface ports. The command will get the registers directly from the address segment objects or from address segments associated with the specified interface pin or interface port objects.

Arguments

`-of_objects` <args> - (Required) The argument value is an object of address segments, interface pins, or external interface ports as returned by the `get_bd_addr_segs`, `get_bd_intf_pins`, or `get_bd_intf_ports` commands. If the object is an address segment, the command will get the registers directly associated with the segment. If the object is either an interface pin or external interface port, the command will find the associated address segments and obtain the registers from those segments.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example creates an external interface pin, assigns an address space, and then gets the registers associated with that address space:

```
% create_bd_cell -type ip -vlnv xilinx.com:ip:axi_gpio:2.0 axi_gpio_0
% make_bd_intf_pins_external  [get_bd_intf_pins axi_gpio_0/S_AXI]
% assign_bd_address
% get_bd_addr_segs
/axi_gpio_0/S_AXI/Reg /S_AXI_0/SEG_axi_gpio_0_Reg
% set seg [get_bd_addr_segs /axi_gpio_0/S_AXI/Reg]
/axi_gpio_0/S_AXI/Reg
% get_bd_regs -of_objects $seg
GPIO_DATA GPIO_TRI GPIO2_DATA GPIO2_TRI GIER IP_IER IP_ISR
%set regs [get_bd_regs -of_objects $seg]
GPIO_DATA GPIO_TRI GPIO2_DATA GPIO2_TRI GIER IP_IER IP_ISR
% set reg [lindex $regs 0]
GPIO_DATA
% report_property $reg
Property       Type    Read-only  Visible   Value
CLASS          string  true       true      bd_reg
DISPLAYNAME    string  true       true      Channel_1_GPIO_DATA
NAME           string  true       true      GPIO_DATA
OFFSET         string  true       true      0x0
```

See Also

- get_bd_addr_segs
- get_bd_intf_pins
- get_bd_intf_ports
- report_property

Send Feedback

get_bel_pins

Get a list of bel_pins. If a design is loaded, gets the constructed site type bels.

Syntax

```
get_bel_pins [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>]
      [-quiet] [-verbose] [<patterns>]
```

Returns

Bel_pin

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-of_objects]	Get the bel_pin of these bels, sites, pins, or nets.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match bel_pin against patterns Default: *

Categories

[Object](#), [XDC](#)

Description

Returns a list of pins on the specified BELs, or matching a specified search pattern.

The default command gets a list of all pins on all BELs on the device.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Send Feedback

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_bel_pins` based on property values on the bel_pins. You can find the properties on an object with the `report_property` or `list_property` commands. Any property/value pair can be used as a filter. In the case of the bel_pin object, "NAME" and "IS_INVERTED" are two of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <args> - (Optional) This option can be used with the `get_bels` command to return the pins of specified BELs.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match bel_pin objects against the specified patterns. The default pattern is the wildcard '*' which gets a list of all bel_pins on the device. More than one search pattern can be specified to find pins based on different search criteria.

Note: You must enclose multiple search patterns in braces, {}, or quotes, "", to present the list as a single element.

Examples

The following example returns the pins of the specified BELs associated with the specified range of sites on the device:

```
get_bel_pins -of_objects [get_bels -of_objects [get_sites \
    -range {SLICE_X0Y0 SLICE_X1Y1}] ]
```

The following example returns the clock enable (CE) pins of all BELs on the device:

```
get_bel_pins *CE
```

See Also

- get_bels
- get_sites
- list_property
- report_property

get_bels

Get a list of bels. If a design is loaded, gets the constructed site type bels.

Syntax

```
get_bels [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>]
    [-include_routing_bels] [-include_routethru_bels] [-quiet] [-verbose]
    [<patterns>]
```

Returns

Bels

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-of_objects]	Get the bels of these slr, tiles, sites, cells, clock_regions, nets or pseudo pips.
[-include_routing_bels]	Also look for and include matching rbels.
[-include_routethru_bels]	Also look for and include routethru bels
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match bels against patterns Default: *

Categories

[Object](#), [XDC](#)

Description

Basic Elements, or BELs, are building blocks of logic, such as flip-flops, LUTs, and carry logic, that make up a SLICE. This command returns a list of BELs on the target part that match a specified search pattern in an open design.

The default command gets a list of all BELs on the device.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Send Feedback

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_bels` based on property values on the BELs. You can find the properties on an object with the `report_property` or `list_property` commands. Any property/value pair can be used as a filter. In the case of the BEL object, "IS_OCCUPIED" and "TYPE" are two of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <args> - (Optional) Get the BELs associated with the specified cells, nets, sites, clock_regions, or slrs.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match BELs against the specified patterns. The default pattern is the wildcard '*' which gets a list of all BELs on the device. More than one search pattern can be specified to find BELs based on different search criteria.

Note: You must enclose multiple search patterns in braces, {}, or quotes, "", to present the list as a single element.

Examples

The following example returns the total number of BELs on the target part:

```
llength [get_bels]
```

The following example returns the BELs associated with the specified site:

```
get_bels -of_objects [get_sites PHASER_IN_PHY_X0Y5]
```

See Also

- get_sites
- list_property
- report_property

Send Feedback

get_board_bus_nets

Gets the list of board bus net objects

Syntax

```
get_board_bus_nets [-regexp] [-nocase] [-all] [-filter <arg>]
    -of_objects <args> [-quiet] [-verbose] [<patterns>...]
```

Returns

List of bus nets in the board.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching
[-all]	Returns all enabled as well as disabled buses
[-filter]	Filter list with expression
-of_objects	Get 'board_bus_net' objects of these types: 'board_bus board_component board_component_pin'.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	match board net names against patterns Default: * Values: The default search pattern is the wildcard *, or .* when -regexp is specified.

Categories

[Object](#), [Board](#)

Description

Gets a list of individual connection bus nets of the specified connection bus object, or the board component or board component pin objects.

The board file, `board.xml` located in the `data/boards` folder of the Vivado Design Suite installation area, stores information regarding board attributes. The board provides a representation of the overall system that the Xilinx device is a part of, and can help define key aspects of the design, such as clock constraints, I/O port assignments, and supported interfaces. You can create custom boards by defining a custom Board Interface file, as described in the *Vivado Design Suite User Guide: System-Level Design Entry (UG895)*.

Connection buses define the connections between the Xilinx device (part0) and other components on the board. Bus nets define individual connections of the connection bus.

Send Feedback

This command returns a list of connection bus nets, or returns an error if it fails.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-all` - (Optional) Return a list of all connection bus nets defined in Board Interface file of the specified object.

`-filter` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_board_bus_nets` based on property values on the bus nets. You can find the properties on an object with the `report_property` or `list_property` commands. Any property/value pair can be used as a filter. In the case of the board bus net object, "BUS", "NAME", and "DELAY" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

Send Feedback

`-of_objects` <args> - (Required) Get the nets of connection buses of the specified board_bus, board_component, or board_component_pin objects.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match board bus nets against the specified search patterns. The default pattern is the wildcard '*' which gets a list of all connection bus nets of the specified objects.

Examples

The following example gets the connection bus nets associated with the specified component of the current board:

```
get_board_bus_nets -of_objects [get_board_components {*iic_main*}]
```

See Also

- current_board_part
- get_board_buses
- get_board_part_interfaces
- get_board_parts
- get_boards

Send Feedback

get_board_buses

Gets the list of board bus objects

Syntax

```
get_board_buses [-regexp] [-nocase] [-all] [-filter <arg>]
     [-of_objects <args>] [-quiet] [-verbose] [<patterns>...]
```

Returns

List of buses in the board.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching
[-all]	Returns all enabled as well as disabled buses
[-filter]	Filter list with expression
[-of_objects]	Get 'board_bus' objects of these types: 'board board_component board_bus_net'.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	match board bus names against patterns Default: * Values: The default search pattern is the wildcard *, or .* when -regexp is specified.

Categories

[Object](), [Board]()

Description

Gets a list of connection buses defined on the current board, as defined in the Board Interface file.

The board file, `board.xml` located in the `data/boards` folder of the Vivado Design Suite installation area, stores information regarding board attributes. The board provides a representation of the overall system that the Xilinx device is a part of, and can help define key aspects of the design, such as clock constraints, I/O port assignments, and supported interfaces. You can create custom boards by defining a custom Board Interface file, as described in the *Vivado Design Suite User Guide: System-Level Design Entry (UG895)*.

Connection buses define the connections between the Xilinx device (part0) and other components on the board.

This command returns a list of buses, or returns an error if it fails.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-all` - (Optional) Return a list of all connection buses defined in Board Interface file of the current board.

`-filter` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_board_buses` based on property values on the buses. You can find the properties on an object with the `report_property` or `list_property` commands. Any property/value pair can be used as a filter. In the case of the board bus object, "COMPONENT_1", "COMPONENT_2", and "DELAY" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <args> - (Optional) Get the connection buses of the specified board, board_component, or board_bus_net objects.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match board buses against the specified search patterns. The default pattern is the wildcard '*' which gets a list of all connection buses defined on the current board.

Examples

The following example gets the connection buses associated with the specified component of the current board:

```
get_board_buses -of_objects [get_board_components sgmii]
```

See Also

- current_board_part
- get_board_part_interfaces
- get_board_parts
- get_boards

Send Feedback

get_board_component_interfaces

Gets the list of interfaces in the board components that implement busdef specified by VLNV

Syntax

```
get_board_component_interfaces [-regexp] [-nocase] [-all] [-filter <arg>]
     [-of_objects <args>] [-quiet] [-verbose] [<patterns>...]
```

Returns

List of bus interfaces.

Usage

Name	Description
`[-regexp]`	Patterns are full regular expressions
`[-nocase]`	Perform case-insensitive matching
`[-all]`	Returns all enabled as well as disabled interfaces
`[-filter]`	Filter list with expression
`[-of_objects]`	Get 'board_component_interface' objects of these types: 'board board_component'.
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`[<patterns>]`	Match Bus Interface names against patterns Default: * Values: The default search pattern is the wildcard *, or .* when -regexp is specified.

Categories

[Object](#), [Board](#)

Description

Gets a list of interfaces defined on the components found on the current board, as defined in the Board Interface file.

The board file, `board.xml` located in the `data/boards` folder of the Vivado Design Suite installation area, stores information regarding board attributes. The board provides a representation of the overall system that the Xilinx device is a part of, and can help define key aspects of the design, such as clock constraints, I/O port assignments, defined components and supported interfaces. You can create custom boards by defining a custom Board Interface file, as described in the *Vivado Design Suite User Guide: System-Level Design Entry (UG895)*.

The component interface defines related groups of signals that are found on a component, or a component mode.

This command returns a list of component interfaces, or returns an error if it fails.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-all` - (Optional) Return a list of all component interfaces defined in the Board Interface file of the current board.

`-filter` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_board_component_interfaces` based on property values on the component interfaces. You can find the properties on an object with the `report_property` or `list_property` commands. Any property/value pair can be used as a filter. In the case of the component interface object, "NAME" and "BUSDEF_NAME" are two of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

Send Feedback

`-of_objects` <args> - (Optional) Get the component interfaces of the specified board, or board_component objects.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match component interfaces against the specified search patterns. The default pattern is the wildcard '*' which gets a list of all component interfaces defined on the current board.

Examples

The following example gets the component interfaces defined in the Board Interface file for the specified board component:

```
get_board_component_interfaces -of_objects [get_board_components *part0*]
```

The following example gets the component interfaces defined in the Board Interface file, and then uses that information to create interfaces in the current project:

```
#Get and Create Interfaces for I/O Ports
foreach x [get_board_component_interfaces] {
create_interface $x }
```

See Also

- current_board_part
- get_board_buses
- get_board_part_interfaces
- get_board_parts
- get_boards

Send Feedback

get_board_component_modes

Gets the list of component mode objects

Syntax

```
get_board_component_modes [-regexp] [-nocase] [-all] [-filter <arg>]
    -of_objects <args> [-quiet] [-verbose] [<patterns>...]
```

Returns

List of component modes in the board.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching
[-all]	Returns all enabled as well as disabled interfaces
[-filter]	Filter list with expression
-of_objects	Get 'board_component_mode' objects of these types: 'board_component'.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	match board component modes against patterns Default: * Values: The default search pattern is the wildcard *, or .* when -regexp is specified.

Categories

[Object](#), [Board](#)

Description

Gets a list of various component modes defined on the current board, as defined in the Board Interface file.

The board file, `board.xml` located in the `data/boards` folder of the Vivado Design Suite installation area, stores information regarding board attributes. The board provides a representation of the overall system that the Xilinx device is a part of, and can help define key aspects of the design, such as clock constraints, I/O port assignments, and supported interfaces. You can create custom boards by defining a custom Board Interface file, as described in the *Vivado Design Suite User Guide: System-Level Design Entry (UG895)*.

The component mode defines various modes of operations that the components on a board may have, and the interfaces and preferred IP of those modes.

This command returns a list of component modes, or returns an error if it fails.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-all` - (Optional) Return a list of all component modes defined in the Board Interface file of the current board.

`-filter` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_board_component_modes` based on property values on the component modes. You can find the properties on an object with the `report_property` or `list_property` commands. Any property/value pair can be used as a filter. In the case of the component mode object, "NAME" and "INTERFACES" are two of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <args> - (Required) Get the component modes of the specified board_components.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match component modes against the specified search patterns. The default pattern is the wildcard '*' which gets a list of all component modes defined on the specified board components.

Examples

The following example gets the component modes defined in the Board Interface file of the specified board:

```
get_board_component_modes -of_objects [get_board_components *part0*]
```

See Also

- current_board
- get_board_buses
- get_board_part_interfaces
- get_board_parts
- get_boards

Send Feedback

get_board_component_pins

Gets the list of board_part pins object

Syntax

```
get_board_component_pins [-regexp] [-nocase] [-filter <arg>]
     -of_objects <args> [-quiet] [-verbose] [<patterns>...]
```

Returns

List of pins in the board_part.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching
[-filter]	Filter list with expression
-of_objects	Get 'board_component_pin' objects of these types: 'board_component board_bus_net'.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	match board component pin names against patterns Default: * Values: The default search pattern is the wildcard *, or .* when -regexp is specified.

Categories

[Object](#), [Board](#)

Description

Gets a list of individual board component pins of the specified board component object of the current_board.

The board file, `board.xml` located in the `data/boards` folder of the Vivado Design Suite installation area, stores information regarding board attributes. The board provides a representation of the overall system that the Xilinx device is a part of, and can help define key aspects of the design, such as clock constraints, I/O port assignments, and supported interfaces. You can create custom boards by defining a custom Board Interface file, as described in the *Vivado Design Suite User Guide: System-Level Design Entry (UG895)*.

Board components define the list of components found on the board defined by the Board Interface file. Component pins enumerate the individual pins of the component.

Send Feedback

This command returns a list of component pin objects, or returns an error if it fails.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_board_component_pins` based on property values on the component pins. You can find the properties on an object with the `report_property` or `list_property` commands. Any property/value pair can be used as a filter. In the case of the board component pin object, "NAME", "IOSTANDARD", and "PIN_INDEX" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <args> - (Required) Get the component pins of the specified board_bus_net, or board_component objects.

Send Feedback

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match board component pins against the specified search patterns. The default pattern is the wildcard '*' which gets a list of all component pins of the specified objects.

Examples

The following example gets the component pins associated with the specified Xilinx device (part0) component object of the current board:

```
get_board_component_pins -of_objects [get_board_components *part0*]
```

See Also

- current_board_part
- get_board_buses
- get_board_components
- get_board_part_interfaces
- get_board_parts
- get_boards

Send Feedback

get_board_components

Get the list of components available in the board

Syntax

```
get_board_components [-regexp] [-nocase] [-all] [-filter <arg>]
      [-of_objects <args>] [-quiet] [-verbose] [<patterns>...]
```

Returns

List of component objects.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching
[-all]	Returns all enabled as well as disabled components
[-filter]	Filter list with expression
[-of_objects]	Get 'board_component' objects of these types: 'board board_bus board_component_pin'.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match component names against patterns Default: * Values: The default search pattern is the wildcard *, or .* when -regexp is specified.

Categories

Object, Board

Description

Gets a list of components defined on the current board, as defined in the Board Interface file.

The board file, `board.xml` located in the `data/boards` folder of the Vivado Design Suite installation area, stores information regarding board attributes. The board provides a representation of the overall system that the Xilinx device is a part of, and can help define key aspects of the design, such as clock constraints, I/O port assignments, and supported interfaces. You can create custom boards by defining a custom Board Interface file, as described in the *Vivado Design Suite User Guide: System-Level Design Entry (UG895)*.

This command returns a list of components, or returns an error if it fails.

Send Feedback

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-all` - (Optional) Return a list of all components defined in Board Interface file of the current board.

`-filter` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_board_components` based on property values on the components. You can find the properties on an object with the `report_property` or `list_property` commands. Any property/value pair can be used as a filter. In the case of the board component object, "COMPONENT_NAME", "TYPE", and "DESCRIPTION" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <args> - (Optional) Get the components of the specified boards, board_buses, or board_component_pins.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match board components against the specified search patterns. The default pattern is the wildcard '*' which gets a list of all board components defined on the current board.

Examples

The following example gets the components defined in the Board Interface file of the specified board:

```
get_board_components -of_objects [get_boards zed]
```

See Also

- current_board_part
- get_board_buses
- get_board_part_interfaces
- get_board_parts
- get_boards

get_board_interface_ports

Gets the list of interface ports object

Syntax

```
get_board_interface_ports [-regexp] [-nocase] [-filter <arg>]
    -of_objects <args> [-quiet] [-verbose] [<patterns>...]
```

Returns

List of ports in the given interface.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching
[-filter]	Filter list with expression
-of_objects	Get 'board_component_pin' objects of these types: 'board_component_interface board_component_pin'.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	match interface port names against patterns Default: * Values: The default search pattern is the wildcard *, or .* when -regexp is specified.

Categories

[Object](), [Board]()

Description

Gets a list of various physical ports assigned to the component interfaces defined on the current board, as defined in the Board Interface file. The interface ports can be returned from component interfaces as specified by the `get_board_component_interfaces` command, or from the component pins returned by `get_board_component_pins`.

The Board Interface file, `board.xml` located in the `data/boards` folder of the Vivado Design Suite installation area, stores information regarding board attributes. The board provides a representation of the overall system that the Xilinx device is a part of, and can help define key aspects of the design, such as clock constraints, I/O port assignments, and supported interfaces. You can create custom boards by defining a custom Board Interface file, as described in the *Vivado Design Suite User Guide: System-Level Design Entry (UG895)*.

In the Board Interface file, a component interface includes a map of the logical ports, that are defined in the interface file, with a physical port that relates to the component pin or pins on the Xilinx device (part0).

This command returns a list of the physical ports of the specified component interface, or returns an error if it fails.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_board_interface_ports` based on property values on the interface ports. You can find the properties on an object with the `report_property` or `list_property` commands. Any property/value pair can be used as a filter. In the case of the interface port object, "LOGICAL_NAME" and "DIRECTION" are two of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Send Feedback

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <args> - (Required) Get the physical interface ports of the specified board component interface, or board component pin objects.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match physical interface ports against the specified search patterns. The default pattern is the wildcard '*' which gets a list of all component interface ports defined on the specified object.

Examples

The following example gets the board interface ports defined in the Board Interface file for the specified board component:

```
get_board_interface_ports -of_objects \
    [get_board_component_interfaces *gmii*]
```

See Also

- current_board_part
- get_board_component_interfaces
- get_board_component_pins
- get_board_components
- get_boards

get_board_ip_preferences

Gets the list of ip preference objects

Syntax

```
get_board_ip_preferences [-regexp] [-nocase] [-filter <arg>]
    -of_objects <args> [-quiet] [-verbose] [<patterns>...]
```

Returns

List of ip preferences for the component.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching
[-filter]	Filter list with expression
-of_objects	Get 'ip_preference' objects of these types: 'board_component_mode board_component_interface'.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	match ip preferences against patterns Default: * Values: The default search pattern is the wildcard *, or .* when -regexp is specified.

Categories

[Object](#), [Board](#)

Description

Gets a list of IP preferences defined on the current board, as defined in the Board Interface file.

The board file, `board.xml` located in the `data/boards` folder of the Vivado Design Suite installation area, stores information regarding board attributes. The board provides a representation of the overall system that the Xilinx device is a part of, and can help define key aspects of the design, such as clock constraints, I/O port assignments, and supported interfaces. You can create custom boards by defining a custom Board Interface file, as described in the *Vivado Design Suite User Guide: System-Level Design Entry (UG895)*.

IP preferences define the preferred IP to connect a component interface to in the Board Interface file.

Send Feedback

This command returns a list of preferred IP for specified component interfaces, or component modes, or returns an error if it fails.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_board_ip_preferences` based on property values on the ip_preference objects. You can find the properties on an object with the `report_property` or `list_property` commands. Any property/value pair can be used as a filter. In the case of the ip_preference object, "IP_NAME" and "IP_VENDOR" are two of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <args> - (Required) Get the preferred IP of the specified board_component_mode or board_component_interface objects.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match preferred IP against the specified search patterns. The default pattern is the wildcard '*' which gets a list of all IP preferences defined for the specified component mode or component interface.

Examples

The following example gets the IP preferences of the specified component interfaces:

```
get_board_ip_preferences -of_objects \
    [get_board_component_interfaces *clock*]
```

See Also

- current_board_part
- get_board_buses
- get_board_part_interfaces
- get_board_parts
- get_boards

Send Feedback

get_board_jumpers

Gets the list of board jumper objects

Syntax

```
get_board_jumpers [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>]
     [-quiet] [-verbose] [<patterns>...]
```

Returns

List of jumpers in the board.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching
[-filter]	Filter list with expression
[-of_objects]	Get 'board_jumper' objects of these types: 'board'.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	match jumper names against patterns Default: * Values: The default search pattern is the wildcard *, or .* when -regexp is specified.

Categories

[Object](), [Board]()

Description

Gets a list of jumpers defined on the current board, as defined in the Board Interface file.

The board file, `board.xml` located in the `data/boards` folder of the Vivado Design Suite installation area, stores information regarding board attributes. The board provides a representation of the overall system that the Xilinx device is a part of, and can help define key aspects of the design, such as clock constraints, I/O port assignments, and supported interfaces. You can create custom boards by defining a custom Board Interface file, as described in the *Vivado Design Suite User Guide: System-Level Design Entry (UG895)*.

The jumpers defined in the board file can be used to enable specific component modes and interfaces of the board.

This command returns a list of jumpers, or returns an error if it fails.

Send Feedback

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_board_jumpers` based on property values on the jumpers. You can find the properties on an object with the `report_property` or `list_property` commands. Any property/value pair can be used as a filter. In the case of the board jumper object, "NAME" and "CURRENT_VALUE" are two of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <args> - (Optional) Get the jumpers of the specified boards.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Send Feedback

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match board jumpers against the specified search patterns. The default pattern is the wildcard '*' which gets a list of all board jumpers defined on the current board.

Examples

The following example gets the jumpers defined in the Board Interface file of the specified board:

```
get_board_jumpers -of_objects [get_boards kc705]
```

See Also

- current_board_part
- get_board_buses
- get_board_part_interfaces
- get_board_parts
- get_boards

Send Feedback

get_board_parameters

Gets the list of board parameter objects

Syntax

```
get_board_parameters [-regexp] [-nocase] [-filter <arg>]
    [-of_objects <args>] [-quiet] [-verbose] [<patterns>...]
```

Returns

List of parameters in the board.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching
[-filter]	Filter list with expression
[-of_objects]	Get 'board_parameter' objects of these types: 'board board_component board_component_interface'.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	match parameter names against patterns Default: * Values: The default search pattern is the wildcard *, or .* when -regexp is specified.

Categories

Object, Board

Description

Gets a list of parameters defined on the current board, as defined in the Board Interface file.

The board file, `board.xml` located in the `data/boards` folder of the Vivado Design Suite installation area, stores information regarding board attributes. The board provides a representation of the overall system that the Xilinx device is a part of, and can help define key aspects of the design, such as clock constraints, I/O port assignments, and supported interfaces. You can create custom boards by defining a custom Board Interface file, as described in the *Vivado Design Suite User Guide: System-Level Design Entry (UG895)*.

The parameters defined in the board file specify custom or user-defined characteristics of the board.

This command returns a list of board parameters, or returns an error if it fails.

Send Feedback

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_board_parameters` based on property values on the parameter objects. You can find the properties on an object with the `report_property` or `list_property` commands. Any property/value pair can be used as a filter. In the case of the board parameter object, "NAME", "VALUE", and "VALUE_TYPE" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <args> - (Optional) Get the parameters of the specified boards, board components, or board component interfaces.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match board parameters against the specified search patterns. The default pattern is the wildcard '*' which gets a list of all board parameters defined on the current board.

Examples

The following example gets the parameters defined in the Board Interface file of the current board:

```
get_board_parameters
```

See Also

- current_board_part
- get_board_buses
- get_board_part_interfaces
- get_board_parts
- get_boards

Send Feedback

get_board_part_interfaces

Gets the list of interfaces in the board_part that implement busdef specified by VLNV

Syntax

```
get_board_part_interfaces [-regexp] [-nocase] [-filter <arg>]
    [-of_objects <args>] [-quiet] [-verbose] [<patterns>...]
```

Returns

List of bus interfaces.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching
[-filter]	Filter list with expression
[-of_objects]	Get 'board_component_interface' objects of these types: 'board board_component'.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match Bus Interface names against patterns Default: * Values: The default search pattern is the wildcard *, or .* when -regexp is specified.

Categories

Object, Board

Description

Gets a list of interfaces defined on the Xilinx device, or current board part as defined by the BOARD_PART property, on the board in use by the current project or open design.

The board file, `board.xml` located in the `data/boards` folder of the Vivado Design Suite installation area, stores information regarding board attributes. The board provides a representation of the overall system that the Xilinx device is a part of, and can help define key aspects of the design, such as clock constraints, I/O port assignments, and supported interfaces. You can create custom boards by defining a custom Board Interface file, as described in the *Vivado Design Suite User Guide: System-Level Design Entry (UG895)*.

The board part provides a representation of the Xilinx device in the context of the board-level system, and is represented by the part0 component in the Board Interface file. The `current_board_part` command returns the board part in use by the current project.

The interfaces defined on the current board part define related groups of signals that can be used to quickly connect the elements of a block design in Vivado IP integrator, or configure IP from the Xilinx IP catalog. The interfaces available on the `current_board_part` can be used to define the interfaces required in an IP subsystem design, using `create_bd_interface_port` or `create_bd_port`. It can also be used to define the interfaces available in the overall FPGA design using `create_interface` and `create_port`.

This command returns a list of available interfaces on the current board part, or returns an error if it fails.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_board_part_interfaces` based on property values on the interfaces. You can find the properties on an object with the `report_property` or `list_property` commands. For example:

```
report_property -all [get_board_part_interfaces ddr3*]
```

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

Send Feedback

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` **<args>** - (Optional) Get the interfaces of the specified board objects, as returned by the `get_boards` command, or board components as returned by `get_board_components`.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match available board part interfaces against the specified search patterns. The default pattern is the wildcard '*' which gets a list of all interfaces available for use in the current project or design. More than one pattern can be specified to find multiple interfaces based on different search criteria.

Note: You must enclose multiple search patterns in braces, {}, or quotes, "", to present the list as a single element.

Examples

The following example gets a list of all interfaces defined on the current board part:

```
join [get_board_part_interfaces] \n
```

See Also

- create_bd_intf_port
- create_interface
- current_board_part
- current_project
- get_boards

Send Feedback

- get_board_components
- get_board_part_pins
- get_board_parts

Send Feedback

www.xilinx.com

get_board_part_pins

Gets the list of board_part pins object

Syntax

```
get_board_part_pins [-regexp] [-nocase] [-filter <arg>]
      [-of_objects <args>] [-quiet] [-verbose] [<patterns>...]
```

Returns

List of pins in the board_part.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching
[-filter]	Filter list with expression
[-of_objects]	Get 'board_component_pin' objects of these types: 'board_component_interface board_interface_port'.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	match board_part pin names against patterns Default: * Values: The default search pattern is the wildcard *, or .* when -regexp is specified.

Categories

[Object](), [Board]()

Description

Gets a list of component pin objects on the current board part in use by the current project or design.

The board file, `board.xml` located in the `data/boards` folder of the Vivado Design Suite installation area, stores information regarding board attributes. The board provides a representation of the overall system that the Xilinx device is a part of, and can help define key aspects of the design, such as clock constraints, I/O port assignments, and supported interfaces. You can create custom boards by defining a custom Board Interface file, as described in the *Vivado Design Suite User Guide: System-Level Design Entry (UG895)*.

The board part provides a representation of the Xilinx device in the context of the board-level system, and is represented by the `part0` component in the Board Interface file. The `current_board_part` command returns the board part in use by the current project.

Send Feedback

The board part pin represents the component pin of an implemented interface on the Xilinx device. The component pin includes properties like LOC, IOSTANDARD, and SLEW. Board part pins can be scalar or vector, so it is always represented as bitwise.

The board part pins can be used to define and place PORTS in the top-level FPGA design, using the `create_port` and `set_property PACKAGE_PIN` commands.

This command returns a list of component pins, or returns an error if it fails.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_board_part_pins` based on property values on the board part pins. You can find the properties on an object with the `report_property` or `list_property` commands. For example:

```
report_property [get_board_part_pins RESET]
```

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <args> - (Optional) Get the pin assignments of the specified board component interface objects, or board interface ports for the current board part.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match available board part pins against the specified search patterns. The default pattern is the wildcard '*' which gets a list of all board part pins available for use in the current project or design. More than one pattern can be specified to find multiple interfaces based on different search criteria.

Note: You must enclose multiple search patterns in braces, {}, or quotes, "", to present the list as a single element.

Examples

This example returns the physical pins of the specified board part interface:

```
get_board_part_pins -of [get_board_part_interfaces push_buttons_5bits]
```

The following example assigns the PACKAGE_PIN and IOSTANDARD properties on the specified port in the current design according to the properties on the leds_8bits pins in the current board:

```
set_property PACKAGE_PIN [get_property LOC \
   [get_board_part_pins leds_8bits_TRI_O[1]]] [get_ports LEDS_n[1]]
set_property IOSTANDARD [get_property IOSTANDARD \
   [get_board_part_pins leds_8bits_TRI_O[1]]] [get_ports LEDS_n[1]]
```

The following example gets a list of board part pins assigned to the leds_8bits board part interface; stores those pins in a Tcl variable $boardPins, and then prints the LOC property for each of those pins:

```
set boardPins [get_board_part_pins -of \
    [get_board_part_interfaces -filter {NAME == led_8bits}]]
foreach pin $boardPins {puts "The location of $pin is: \
    [get_property LOC $pin]"}
The location of leds_8bits_tri_o[0] is: AB8
The location of leds_8bits_tri_o[1] is: AA8
The location of leds_8bits_tri_o[2] is: AC9
The location of leds_8bits_tri_o[3] is: AB9
The location of leds_8bits_tri_o[4] is: AE26
The location of leds_8bits_tri_o[5] is: G19
The location of leds_8bits_tri_o[6] is: E18
The location of leds_8bits_tri_o[7] is: F16
```

See Also

- create_bd_intf_port
- create_interface
- current_board_part
- current_project
- get_board_part_interfaces
- get_board_parts

get_board_parts

Get the list of board_part available in the project

Syntax

```
get_board_parts [-regexp] [-nocase] [-latest_file_version]
    [-latest_hw_revision] [-filter <arg>] [-quiet] [-verbose]
    [<patterns>...]
```

Returns

List of board_part objects.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching
[-latest_file_version]	Show only latest board parts by file version
[-latest_hw_revision]	Show only latest board parts by board revision
[-filter]	Filter list with expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match Board Part names against patterns Default: * Values: The default search pattern is the wildcard *, or .* when -regexp is specified.

Categories

[Object](), [Project](), [Board]()

Description

Gets a list of available board parts in the board repository, as defined by the Board Interface files available for use by the current project or design.

The board file, `board.xml` located in the `data/boards` folder of the Vivado Design Suite installation area, stores information regarding board attributes. The board provides a representation of the overall system that the Xilinx device is a part of, and can help define key aspects of the design, such as clock constraints, I/O port assignments, and supported interfaces. You can create custom boards by defining a custom Board Interface file, as described in the *Vivado Design Suite User Guide: System-Level Design Entry (UG895)*.

Send Feedback

The board part provides a representation of the Xilinx device in the context of the board-level system, and is represented by the `part0` component in the Board Interface file. The `current_board_part` command returns the board part in use by the current project. Refer to the `current_board_part` command for the different methods of defining the board in use.

This command returns the list of available Xilinx devices (part0) in the Board Interface files defined in the current board repository, or returns an error if it fails.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-latest_file_version` - (Optional) Return the board parts defined in the latest version of the Board Interface file. There can be multiple versions of the Board Interface file. This option returns the board parts in the latest version only. Refer to the *Vivado Design Suite User Guide: System-Level Design Entry (UG895)* for more information on the Board Interface file.

`-latest_hw_revision` - (Optional) Return the board parts defined in the latest compatible hardware revision of the board represented in the Board Interface file. The board Interface file can represent multiple compatible revisions of boards. This option only returns the latest revision.

`-filter` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_board_parts` based on property values on the board parts. You can find the properties on an object with the `report_property` or `list_property` commands. Any property/value pair can be used as a filter. In the case of the board part object, "NAME", "PART_NAME", and "BOARD_NAME" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Send Feedback

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match board parts against the specified search patterns. The default pattern is the wildcard '*' which gets a list of all board parts available for use in the project. More than one pattern can be specified to find multiple boards based on different search criteria.

Note: You must enclose multiple search patterns in braces, {}, or quotes, "", to present the list as a single element.

Examples

The following example returns the board parts matching the specified filter search pattern:

```
get_board_parts -filter {BOARD_NAME=~z*}
```

The following example returns all board parts matching the specified search patterns:

```
get_board_parts {*av* *kc*}
```

See Also

- current_board_part
- get_board_part_interfaces
- get_board_part_pins

- list_property
- report_property
- set_property

get_boards

Get the list of boards available in the project

Syntax

```
get_boards [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>]
     [-quiet] [-verbose] [<patterns>...]
```

Returns

List of board objects.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching
[-filter]	Filter list with expression
[-of_objects]	Get 'board' objects of these types: 'board_component'.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match Board names against patterns Default: * Values: The default search pattern is the wildcard *, or .* when -regexp is specified.

Categories

Object, Project, Board

Description

Gets a list of evaluation boards available for use by the current project.

The board file, `board.xml` located in the `data/boards` folder of the Vivado Design Suite installation area, stores information regarding board attributes. The board provides a representation of the overall system that the Xilinx device is a part of, and can help define key aspects of the design, such as clock constraints, I/O port assignments, and supported interfaces. You can create custom boards by defining a custom Board Interface file, as described in the *Vivado Design Suite User Guide: System-Level Design Entry (UG895)*.

The board in use by the project is returned by the `current_board_part` command.

The board can be specified:

- When the project is created by selecting Boards from the Default Part dialog box.

- By setting the BOARD property on the current project as shown in the example.

- By selecting the Project Device in the Settings dialog box in an open project in the Vivado IDE.

Refer to the *Vivado Design Suite User Guide: System-Level Design Entry (UG895)* for information on creating projects, and on configuring project settings.

> **IMPORTANT!** *When you specify the board with the* `set_property` *command, the target part is also changed to match the part required by the specified BOARD property.*

This command returns a list of boards that are available for use by the current project, or returns an error if it fails.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_boards` based on property values on the boards. You can find the properties on an object with the `report_property` or `list_property` commands. Any property/value pair can be used as a filter. In the case of the board object, "NAME", "DEVICE", and "FAMILY" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

Send Feedback

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <args> - (Optional) Get the boards of the specified board_component objects as returned by `get_board_components`.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match boards against the specified search patterns. The default pattern is the wildcard '*' which gets a list of all boards available for use in the project. More than one pattern can be specified to find multiple boards based on different search criteria.

Note: You must enclose multiple search patterns in braces, {}, or quotes, "", to present the list as a single element.

Examples

The following example reports the properties of the specified evaluation board:

```
report_property [get_boards -filter {LIBRARY_NAME==artix7}]
```

The following example returns all boards matching the specified search patterns:

```
get_boards {*ar* *kc*}
```

See Also

- current_board_part
- get_board_interface_ports
- get_board_part_pins
- list_property

- report_property
- set_property

get_cdc_violations

Get a list of CDC violations from a previous report_cdc run

Syntax

```
get_cdc_violations [-name <arg>] [-regexp] [-filter <arg>] [-nocase]
    [-quiet] [-verbose] [<patterns>]
```

Returns

List of CDC violation objects.

Usage

Name	Description
[-name]	Get the results with this name
[-regexp]	Patterns are full regular expressions
[-filter]	Filter list with expression
[-nocase]	Perform case-insensitive matching (valid only when -regexp specified)
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match CDC violations against patterns Default: * Values: The default search pattern is the wildcard *, or .* when -regexp is specified.

Categories

[Object]

Description

Gets a list of violation objects found in the design when the `report_cdc` command is run. The properties of individual violation objects can be queried using `report_property` or `list_property` commands for details of the violation.

Violation objects are associated with the clock-domain crossing paths in the current design. The design objects associated with a methodology violation object can be obtained using the `-of_objects` option of the appropriate `get_*` command, such as `get_cells`, or `get_nets` for instance.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Arguments

`-name <arg>` - (Optional) Get the violations associated with the named CDC result set. In this case the `report_cdc` command must also have been run with the `-name` option.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_cdc_violations` based on property values on the violations. You can find the properties on an object with the `report_property` or `list_property` commands.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Send Feedback

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match violations against the specified patterns. The default pattern is the wildcard '*' which gets all violations. More than one pattern can be specified to find multiple violations based on different search criteria.

Note: You must enclose multiple search patterns in braces, {}, or quotes, "", to present the list as a single element.

Examples

The following example reports the CDC violations found in the current design, then returns a list of all those violations:

```
report_cdc
get_cdc_violations
```

The following example generates list of violations in the named CDC report, and then gets the pins associated with any violations found:

```
report_cdc -name cdc_1
get_pins -of_objects [get_cdc_violations -name cdc_1]
```

See Also

- list_property
- report_cdc
- report_property

Send Feedback

get_cells

Get a list of cells in the current design

Syntax

```
get_cells [-hsc <arg>] [-hierarchical] [-regexp] [-nocase] [-filter <arg>]
    [-of_objects <args>] [-match_style <arg>] [-include_replicated_objects]
    [-quiet] [-verbose] [<patterns>]
```

Returns

List of cell objects.

Usage

Name	Description
[-hsc]	Hierarchy separator Default: /
[-hierarchical]	Search level-by-level in current instance
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching (valid only when -regexp specified)
[-filter]	Filter list with expression
[-of_objects]	Get cells of these pins, timing paths, nets, bels, clock regions, sites, or drc violations
[-match_style]	Style of pattern matching Default: sdc Values: ucf, sdc
[-include_replicated_objects]	Include replicated objects when searching for patterns. This option is valid only when patterns is specified.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match cell names against patterns Default: *

Categories

[SDC](), [XDC](), [Object]()

Description

Gets a list of cell objects in the current design that match a specified search pattern. The default command returns a list of all cells in the current_instance of the open design, as specified by the `current_instance` command.

You can use the `-hierarchical` option to extract cells from the hierarchy of the current design.

Send Feedback

The `get_cells` command also includes an option to get all replicated cells that are added to a design during physical optimization, or `phys_opt_design`. The use of the `-include_replicated_objects` option returns the replicated cells of an object when the original cell is returned. You can use this option to ensure that constraints or properties that are applied to a cell are also applied to its replicated cells.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Arguments

`-hsc <arg>` - (Optional) Set the hierarchy separator. The default hierarchy separator is '/'.

`-hierarchical` - (Optional) Get cells from all levels of the design hierarchy starting from the level of the current_instance, or from the top of the current design. Without this argument, the command will only get cells from the current_instance of the design hierarchy. When using `-hierarchical`, the search pattern should not contain a hierarchy separator because the search pattern is applied at each level of the hierarchy, not to the full hierarchical cell name. For instance, searching for U1/* searches each level of the hierarchy for instances with U1/ in the name. This may not return the intended results. This is illustrated in the examples below.

> **IMPORTANT!** *When used with `-regexp`, the specified search string is matched against the full hierarchical name, and the U1/* search pattern will work as intended.*

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter <args>` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_cells` based on property values on the cells. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the "cell" object, "IS_PARTITION", "IS_PRIMITIVE" and "IS_LOC_FIXED" are some of the properties that can be used to filter results.

Send Feedback

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <arg>` - (Optional) Get the cells connected to the specified pins, timing paths, nets, bels, clock regions, sites or DRC violation objects.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-match_style` - (Optional) Indicates that the search pattern matches UCF constraints or SDC constraints. The default is SDC.

`-include_replicated_objects` - (Optional) Include cells that have been added through replication during optimizations. This option is valid only when specified with <patterns>, and returns replicated cell instances that match the pattern. As a default, the `get_cells` command does not return replicated cells.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Send Feedback

<patterns> - (Optional) Match cells against the specified patterns. The default pattern is the wildcard '*' which gets a list of all cells in the project. More than one pattern can be specified to find multiple cells based on different search criteria.

Note: You must enclose multiple search patterns in braces, {}, or quotes, "", to present the list as a single element.

Examples

The following example uses regular expression to return cells in the BFT example design that match the filter expressions for NAME and REF_NAME. In the first command the NAME is specified in the search pattern, while the second command filters on the NAME property. These commands return the same results:

```
get_cells -regexp -filter { REF_NAME =~ FD.* } .*validFor.*
get_cells -regexp -filter { NAME =~ .*validFor.* && REF_NAME =~ FD.* }
```

The following example searches all levels of the hierarchy for cells beginning with cpu, or fft, and joins each cell returned with the newline character to put it on a separate line:

```
join [get_cells -hier {cpu* fft*}] \n
```

This example gets a list of properties and property values attached to the second object of the list returned by get_cells:

```
report_property [lindex [get_cells] 1]
```

Note: If there are no cells matching the pattern you will get a warning.

This example prints a list of the library cells instantiated into the design at all levels of the hierarchy, sorting the list for unique names so that each cell is only printed one time:

```
foreach cell [lsort -unique [get_property LIB_CELL [get_cells -hier \
-filter {IS_PRIMITIVE==1}]]] {puts $cell}
```

The following example demonstrates the effect of `-hierarchical` searches, without and with `-regexp`:

```
get_cells -hierarchical *mmcm*
   mmcm_replicator_inst_1
   mmcm_replicator_inst_1/mmcm_stage[0].mmcm_channel[0].mmcm
get_cells -hierarchical -regexp .*mmcm.*
   mmcm_replicator_inst_1
   mmcm_replicator_inst_1/mmcm_stage[0].mmcm_channel[0].mmcm
   mmcm_replicator_inst_1/mmcm_stage[0].mmcm_channel[0].mmcm/GND
   mmcm_replicator_inst_1/mmcm_stage[0].mmcm_channel[0].mmcm/MMCM_Base
```

Note: The last two cells (GND and MMCM_Base) were not returned in the first example (without `-regexp`) because the cell names do not match the search pattern, as it is applied to each level of the hierarchy.

The following example runs the `report_drc` command on the current design, and returns any cells associated with DRC violations:

```
report_drc -name drc_1
get_cells -of_objects [get_drc_violations]
```

See Also

- current_instance
- get_drc_violations
- get_lib_cells
- get_nets
- get_pins
- list_property
- phys_opt_design
- report_drc
- report_property

Send Feedback

get_cfgmem_parts

Get a list of cfgmem_parts available in the software

Syntax

```
get_cfgmem_parts [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>]
    [-quiet] [-verbose] [<patterns>]
```

Returns

List of cfgmem_part objects.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-of_objects]	Get 'cfgmem_part' objects of these types: 'part hw_device'.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match the 'cfgmem_part' objects against patterns. Default: *

Categories

[Hardware](#), [Object](#)

Description

Get a list of configuration flash memory devices supported by the Vivado Design Suite or Vivado Lab Edition.

Xilinx® FPGAs can be configured from an external nonvolatile memory device, or they can be configured by an external smart source, such as a microprocessor, DSP processor, microcontroller, PC, or board tester. The two configuration datapaths include a serial datapath that is used to minimize the device pin requirements for configuration, and a parallel 8-bit, 16-bit, or 32-bit datapath used for higher performance or link to industry-standard interfaces ideal for external data sources like processors, or x8- or x16-parallel flash memory.

The process whereby the design specific data is loaded or programmed into the Xilinx FPGA is called configuration. The `create_hw_cfgmem` command defines a flash memory device used for configuring and booting the hardware device.

Send Feedback

After the hw_cfgmem object is created, and associated with the hw_device, the configuration memory can be programmed with the bitstream and other data from a memory configuration file created with the `write_cfgmem` command. The hw_cfgmem object is programmed using the `program_hw_cfgmem` command.

The configuration memory part can be used to define the hardware configuration memory in the Hardware Manager of the Vivado Design Suite, to enable programming and debugging the design in the FPGA hardware. Use the `create_hw_cfgmem` command to define the configuration memory for use with the current hardware device.

This command returns a list of cfgmem_part objects that match the specified search criteria, or returns an error if it fails.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_cfgmem_parts` based on property values on the cfgmem_parts objects. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the "cfgmem_part" object, "COMPATIBLE_PARTS", "DATA_WIDTH", and "MEM_DENSITY" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

Send Feedback

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <arg>` - (Optional) Get the cfgmem_parts of the specified part or hw_device objects.

> ⭐ **IMPORTANT!** *The Vivado Lab Edition does not support part objects, and only supports hw_device objects. The part or hw_device objects must be specified as objects returned by* `get_parts`, `get_hw_devices`, *or* `current_hw_device` *commands, rather than specified by name.*

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match cfgmem_parts against the specified patterns. The default pattern is the wildcard '*' which gets a list of all cfgmem_parts currently available in the Vivado Design Suite.

Example

The following example gets the cfgmem parts compatible with the current hw_device:

```
get_cfgmem_parts -of_objects [current_hw_device]
```

The following example gets the cfgmem_part compatible with the target part, filtered to return only the cfgmem_parts with more than a specified amount of memory:

```
get_cfgmem_parts -of [get_parts [get_property PART [current_hw_device]]] \
-filter {MEM_DENSITY > 128}
```

See Also

- create_hw_cfgmem

- current_hw_device
- delete_hw_cfgmem
- get_hw_devices
- get_parts
- get_property
- program_hw_cfgmem
- set_property
- write_cfgmem

get_clock_regions

Get the clock regions for the current device.

Syntax

```
get_clock_regions [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>]
      [-quiet] [-verbose] [<patterns>]
```

Returns

Clock_regions

Usage

Name	Description
[-regexp]	Patterns are full regular expressions.
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified).
[-filter]	Filter list with expression
[-of_objects]	Get the clock_regions of these tiles, sites, slrs, cells, or package bank
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match objects' name against patterns. Default: *

Categories

Object

Description

Gets a list of clock regions on the target part that match a specified search pattern. The default command gets a list of all clock regions on the device in an open design.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Send Feedback

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_clock_regions` based on property values on the clock regions. You can find the properties on an object with the `report_property` or `list_property` commands. Any property/value pair can be used as a filter. In the case of the clock region object, "COLUMN_INDEX", "HIGH_X", and "LOW_X" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <args> - (Optional) Get the clock regions of the SLRs that the clock region is part of using the `get_slrs` command. Or get the clock_regions that the specified tiles, device sites, I/O banks, or cells are assigned to.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match clock regions against the specified patterns. The default pattern is the wildcard '*' which gets a list of all clock regions on the device. More than one search pattern can be specified to find clock regions based on different search criteria.

Note: You must enclose multiple search patterns in braces, {}, or quotes, "", to present the list as a single element.

Examples

The following example returns the clock regions matching the search pattern:

```
get_clock_regions X0*
```

The following example returns the clock regions filtered by the specified property:

```
get_clock_regions -filter {LOW_X==0}
```

Note: These two examples return the same set of clock regions.

The following example returns the clock regions that the specified ILA debug core trigger is assigned to, or placed in:

```
get_clock_regions -of_objects [get_cells -hierarchical basic_trigger*]
```

See Also

- get_sites
- list_property
- report_property

get_clocks

Get a list of clocks in the current design

Syntax

```
get_clocks [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>]
    [-match_style <arg>] [-include_generated_clocks] [-quiet] [-verbose]
    [<patterns>]
```

Returns

List of clocks.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching (valid only when -regexp specified)
[-filter]	Filter list with expression
[-of_objects]	Get clocks of these pins, nets, or cells
[-match_style]	Style of pattern matching, valid values are ucf, sdc Default: sdc
[-include_generated_clocks]	Include auto-inferred/generated_clocks also.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match clock names against patterns Default: *

Categories

SDC, XDC, Object

Description

Gets a list of clocks in the current design that match a specified search pattern. The default command gets a list of all clocks in the design, like the `all_clocks` command.

Clocks can be created using the `create_clock` or the `create_generated_clock` commands, or can be automatically generated by the tool, at the output of an MMCM for instance.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Send Feedback

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_clocks` based on property values on the clocks. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the clock object, "PERIOD", "WAVEFORM", and "IS_GENERATED" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <args> - (Optional) Get the clocks connected to the specified cell, pin, port, or net objects.

Send Feedback

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-match_style` - (Optional) Indicates that the search pattern matches UCF constraints or SDC constraints. The default is SDC.

`-include_generated_clocks` - (Optional) Returns all clocks, including generated clocks that match the specified pattern. This argument should be used when clock <patterns> are specified in order to return generated clocks of the specified master clocks.

Note: You can get just the generated clocks with the `get_generated_clocks` command.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match clocks against the specified patterns. The default pattern is the wildcard '*' which gets all clocks in the project. More than one pattern can be specified to find multiple clocks based on different search criteria.

Note: You must enclose multiple search patterns in braces, {}, or quotes, "", to present the list as a single element.

Examples

The following example gets a list of clocks matching the various search patterns:

```
get_clocks {*clock *ck *Clk}
```

Note: If there are no clocks matching the pattern you will get a warning.

The following example gets the master clock object, and all generated clocks derived from that clock:

```
get_clocks -include_generated_clocks wbClk
```

The following example gets all properties and property values attached to the specified clock:

```
report_property -all [get_clocks wbClk]
```

Send Feedback

See Also

- all_clocks
- create_clock
- create_generated_clock
- get_generated_clocks
- list_property
- report_property

get_cluster_configurations

Returns all cluster configuration objects

Syntax

```
get_cluster_configurations [-filter <arg>] [-regexp] [-nocase] [-quiet]
    [-verbose] [<patterns>]
```

Usage

Name	Description
[-filter]	Filter list with expression
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching (valid only when -regexp specified)
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match cluster configuration against patterns Default: *

Categories

[cluster_configuration](#)

Description

Gets a list of cluster configurations that includes Vivado default cluster configurations and cluster configurations created using the `create_cluster_configuration` command.

Vivado supports the following cluster management tools.

- Load Sharing Facility (LSF)

- Sun Grid Engine (SGE)

- Simple Linux Utility For Resource Management (SLURM)

Arguments

`-filter` <args> - (Optional) Filter the results list with the specified expression. The -filter argument filters the list of objects returned by `get_cluster_configurations` based on property values on the cluster configurations. You can find the properties on an object with the `report_property` or `list_property` commands. Any property/value pair can be used as a filter. In the case of the cluster configurations object, "TYPE" and "IS_DEFAULT" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets a default cluster configurations with name lsf:

```
get_cluster_configurations * -filter {IS_DEFAULT == 1 && NAME =~ "lsf"}
```

Boolean (bool) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_DEFAULT}
```

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<patterns>` - (Optional) Match cluster configuration names against the specified patterns. The default pattern is the wildcard * which gets a list of all defined cluster configurations. More than one pattern can be specified to find multiple cluster configurations based on different search criteria.

Note: You must enclose multiple search patterns in braces, {}, or quotes, "", to present the list as a single element.

Examples

The following example gets the list of all cluster configurations.

```
get_cluster_configurations
```

The following example gets the cluster configuration with name 'lsf'.

```
get_cluster_configurations lsf
```

The following example gets the list of all default cluster configurations.

```
get_cluster_configurations -filter {IS_DEFAULT}
```

See Also

- create_cluster_configuration
- remove_cluster_configurations
- validate_cluster_configurations

get_constant_paths

Get constant paths

Syntax

```
get_constant_paths [-max_paths_per_endpoint <arg>] [-filter <arg>]
     [-regexp] [-nocase] [-quiet] [-verbose] <objects>
```

Usage

Name	Description
`[-max_paths_per_endpoint]`	number of branches per endpoint to report traces for Default: 1
`[-filter]`	Filter list with expression
`[-regexp]`	Patterns specified in filter are full regular expressions
`[-nocase]`	Perform case-insensitive matching for patterns specified in filter (valid only when -regexp specified)
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<objects>`	endpoint pins or instances

Categories

[Object](), [Timing]()

Description

Get constant paths objects that meet the specified criteria.

Arguments

`-max_paths_per_endpoint` – Specifies maximum number of paths to report per constant endpoint object.

Note: By default, 1 path per endpoint has been reported.

`-filter <args>` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by the search pattern, based on specified property values.

`-regexp` - (Optional) Specifies that the search `<patterns>` are written as regular expressions. Both search `<patterns>` and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string.

Send Feedback

Note: You can add ".*" to the beginning or end of a search string to widen the search to include a substring.

-nocase - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of -regexp only.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

<objects> - (Required) design cell(s) or pin(s) object list to report

Example

Gets constant paths of the specified pin.

```
get_constant_paths [get_pins cpuEngine/or1200_cpu/or1200_sprs/esr/D]
```

Gets constant paths of the specified cell.

```
report_constant_paths [get_cells cpuEngine/or1200_cpu/or1200_sprs/esr]
```

See also

- report_constant_paths
- get_path_groups
- group_path
- place_design
- report_timing_summary
- route_design
- set_clock_groups
- set_false_path

Send Feedback

get_dashboard_gadgets

Create the Project summary dashboard

Syntax

```
get_dashboard_gadgets [-quiet] [-verbose] [<patterns>...]
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match gadget names against patterns Default: * Values: The default search pattern is the wildcard *

Categories

[Object](), [Project]()

Description

Gets a list of the dashboard gadgets in the current project.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match dashboard gadget names against the specified patterns. The default pattern is the wildcard '*' which gets all dashboard gadgets. More than one pattern can be specified to find dashboard gadgets based on multiple search criteria.

Examples

The following example returns the dashboard gadgets in the current project:

```
get_dashboard_gadgets
```

Send Feedback

See Also

- create_dashboard_gadget
- delete_dashboard_gadgets
- move_dashboard_gadget

get_debug_cores

Get a list of debug cores in the current design

Syntax

```
get_debug_cores [-filter <arg>] [-of_objects <args>] [-regexp] [-nocase]
    [-quiet] [-verbose] [<patterns>]
```

Returns

List of debug_core objects.

Usage

Name	Description
[-filter]	Filter list with expression
[-of_objects]	Get cores of these debug ports or nets
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching (valid only when -regexp specified)
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match debug cores against patterns Default: *

Categories

Object, Debug, XDC

Description

Gets a list of Vivado Lab Edition debug cores in the current project that match a specified search pattern. The default command gets a list of all debug cores in the project.

Debug cores are added to the project with the `create_debug_core` command. When a ILA debug core (labtools_ila_v3) is added to the project, it is contained within a Debug Hub core (labtools_xsdbmasterlib_v2), and includes a CLK port and a PROBE port as a default. Additional ports can be added to the debug core with the use of the `create_debug_port` command.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Arguments

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_debug_cores` based on property values on the debug cores. You can find the properties on an object with the `report_property` or `list_property` commands.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <args> - (Optional) Get the ILA debug cores associated with the specified debug ports, or nets.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match debug cores against the specified patterns. The default pattern is the wildcard '*' which gets all debug cores. More than one pattern can be specified to find multiple debug cores based on different search criteria.

Note: You must enclose multiple search patterns in braces, {}, or quotes, "", to present the list as a single element.

Examples

The following command gets a list of the Vivado Lab Edition debug cores in the current project:

```
get_debug_cores
```

Note: A Debug Hub core is returned as one of the debug cores in the project. You cannot directly create this core, but it is automatically added by the tool when you add any ILA cores to the project.

The following example gets the properties of the specified debug core:

```
report_property [get_debug_cores myCore]
```

See Also

- create_debug_core
- create_debug_port
- get_debug_ports
- list_property
- report_property
- set_property

get_debug_ports

Get a list of debug ports in the current design

Syntax

```
get_debug_ports [-filter <arg>] [-of_objects <args>] [-regexp] [-nocase]
    [-quiet] [-verbose] [<patterns>]
```

Returns

List of debug_port objects.

Usage

Name	Description
[-filter]	Filter list with expression
[-of_objects]	Get ports of these debug cores
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching (valid only when -regexp specified)
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match debug ports against patterns Default: *

Categories

Object, Debug, XDC

Description

Gets a list of ports defined on ILA debug cores in the current project that match a specified search pattern. The default command gets a list of all debug ports in the project.

Debug ports are defined when ILA debug cores are created with the `create_debug_core` command. Ports can be added to existing debug cores with the `create_debug_port` command.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Send Feedback

Arguments

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_debug_ports` based on property values on the debug ports. You can find the properties on an object with the `report_property` or `list_property` commands. Any property/value pair can be used as a filter. In the case of the debug_port object, "PORT_WIDTH", and "MATCH_TYPE" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <args> - (Optional) Get the ChipScope debug ports associated with the specified debug cores.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match debug ports against the specified patterns. The default pattern is the wildcard '*' which gets all debug ports. More than one pattern can be specified to find multiple debug ports based on different search criteria.

Note: You must enclose multiple search patterns in braces, {}, or quotes, "", to present the list as a single element.

Examples

The following command gets a list of the ports from the ILA debug cores in the current project, with a PORT_WIDTH property of 8:

```
get_debug_ports -filter {PORT_WIDTH==8}
```

The following example gets the properties attached to the specified debug port:

```
report_property [get_debug_ports myCore/PROBE1]
```

Note: The debug port is defined by the core_name/port_name combination.

See Also

- create_debug_core
- create_debug_port
- list_property
- report_property

Send Feedback

get_designs

Get a list of designs in the current project

Syntax

```
get_designs [-regexp] [-nocase] [-filter <arg>] [-quiet] [-verbose]
    [<patterns>]
```

Returns

List of design objects.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching (valid only when -regexp specified)
[-filter]	Filter list with expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match design names against patterns Default: *

Categories

Object

Description

Gets a list of open designs in the current project that match a specified search pattern. The default command gets a list of all open designs in the project.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_designs` based on property values on the designs. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the "design" object, "CONSTRSET", and "PART" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match designs against the specified patterns. The default pattern is the wildcard '*' which gets all designs. More than one pattern can be specified to find multiple designs based on different search criteria.

Examples

The following example gets all open designs in the current project:

```
get_designs
```

The following example gets the assigned properties of an open design matching the search pattern:

```
report_property [get_designs r*]
```

Note: If there are no designs matching the pattern you will get a warning.

See Also

- report_property

Send Feedback

get_drc_checks

Get a list of DRC rule check objects

Syntax

```
get_drc_checks [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
     [-abbrev <arg>] [-ruledecks <args>] [-quiet] [-verbose] [<patterns>]
```

Returns

List of DRC rule_check objects.

Usage

Name	Description
[-of_objects]	Get 'rule_check' objects of these types: 'drc_ruledeck'.
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-abbrev]	Get the largest ID for this abbrev
[-ruledecks]	Containers of Report DRC rule checks Default: default
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match the 'rule_check' objects against patterns. Default: *

Categories

DRC, Object

Description

Gets a list of the currently defined DRC checks. This list includes both factory defined design rule checks, and user-defined checks created by the `create_drc_check` command.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Arguments

`-of_objects` <args> - (Optional) Get the design rule checks associated with a rule deck object. The ruledeck object must be specified by the `get_drc_ruledecks` command.

Note: `-of_objects` cannot be used with a search `<pattern>`.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of rule checks returned by `get_drc_checks` based on property values on the rule checks. You can find the properties on an object with the `report_property` or `list_property` commands.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-abbrev` <arg> - (Optional) Get the design rule checks associated with the specified rule name or abbreviation.

`-ruledecks` <args> - (Optional) Get the design rule checks associated with the specified rule decks.

Send Feedback

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match design rule checks against the specified patterns. The default pattern is the wildcard '*' which gets all rule checks. More than one pattern can be specified to find multiple rule checks based on different search criteria.

Note: You must enclose multiple search patterns in braces, {}, or quotes, "", to present the list as a single element.

Examples

The following command gets a list of all AVAL design rule checks:

```
get_drc_checks AVAL*
```

The following example gets the checks associated with the specified rule deck:

```
get_drc_checks -of_objects [get_drc_ruledecks placer_checks]
```

See Also

- create_drc_check
- get_drc_ruledecks
- list_property
- report_drc
- report_property

get_drc_ruledecks

Get a list of DRC rule deck objects

Syntax

```
get_drc_ruledecks [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
     [-quiet] [-verbose] [<patterns>]
```

Returns

Drc_ruledeck

Usage

Name	Description
[-of_objects]	Get 'drc_ruledeck' objects of these types: 'rule_check'.
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match the 'drc_ruledeck' objects against patterns. Default: *

Categories

[DRC](), [Object]()

Description

Gets a list of currently defined rule decks for use with the `report_drc` command.

A rule deck is a collection of design rule checks grouped for convenience, to be run at different stages of the FPGA design flow, such as during I/O planning or placement. The tool comes with a set of factory defined rule decks, but you can also create new user-defined rule decks with the `create_drc_ruledeck` command, and add checks to the rule deck using the `add_drc_checks` command.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Arguments

`-of_objects` <args> - (Optional) Get the DRC ruledeck object of the associated rule check objects.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_drc_ruledecks` based on property values on the rule decks. You can find the properties on an object with the `report_property` or `list_property` commands.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Send Feedback

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match rule decks against the specified patterns. The default pattern is the wildcard '*' which gets all rule decks. More than one pattern can be specified to find multiple rule decks based on different search criteria.

Note: You must enclose multiple search patterns in braces, {}, or quotes, "", to present the list as a single element.

Examples

The following example gets a list of rule decks defined in the current project:

```
get_drc_ruledecks
```

The following example lists each of the checks associated with the placer_checks rule deck on a separate line:

```
foreach rule [get_drc_checks -of_objects \
    [get_drc_ruledecks placer_checks]] {puts $rule}
```

See Also

- add_drc_checks
- create_drc_ruledeck
- list_property
- report_drc
- report_property

get_drc_violations

Get a list of DRC violations from a previous report_drc run

Syntax

```
get_drc_violations [-name <arg>] [-regexp] [-filter <arg>] [-nocase]
    [-quiet] [-verbose] [<patterns>]
```

Returns

List of DRC violation objects.

Usage

Name	Description
[-name]	Get the results with this name
[-regexp]	Patterns are full regular expressions
[-filter]	Filter list with expression
[-nocase]	Perform case-insensitive matching (valid only when -regexp specified)
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match drc_violations against patterns Default: * Values: The default search pattern is the wildcard *, or .* when -regexp is specified.

Categories

[DRC](#), [Object](#)

Description

Gets a list of violation objects found in the design when the `report_drc` command is run. Violation objects are created at the time DRC is run, either by the internal design rule checks provided by the Vivado tools, or created by the `create_drc_violation` command in user-defined DRC checks. The properties of individual violation objects can be queried using `report_property` or `list_property` commands for details of the violation.

Violation objects are associated with the cells, nets, pins, or ports in the current design, or sites on the current device. The design objects associated with a DRC violation object can be obtained using the `-of_objects` option of the appropriate `get_*` command, such as `get_cells`, or `get_nets` for instance.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Arguments

`-name <arg>` - (Optional) Get the violations associated with the named DRC result set. In this case the `report_drc` command must have been run with the `-name` option.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter <args>` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_drc_violations` based on property values on the violations. You can find the properties on an object with the `report_property` or `list_property` commands.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

Send Feedback

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match violations against the specified patterns. The default pattern is the wildcard '*' which gets all violations. More than one pattern can be specified to find multiple violations based on different search criteria.

Note: You must enclose multiple search patterns in braces, {}, or quotes, "", to present the list as a single element.

Examples

The following example reports the DRC violations found in the current design, then returns a list of all those violations:

```
report_drc
get_drc_violations
```

The following example gets the properties of the specified DRC violation:

```
report_property [lindex [get_drc_violations] 0]
```

The following example returns the list of violations in the specified DRC report of the current design, and then returns the ports associated with any violations of the unspecified I/O Standard rule (NSTD):

```
get_drc_violations -name drc_1
get_ports -of_objects [get_drc_violations -name drc_1 NSTD*]
```

See Also

- create_drc_check
- create_drc_violation
- get_cells
- get_nets
- get_pins
- get_ports
- get_sites
- report_drc

get_example_designs

Get a list of example designs

Syntax

```
get_example_designs [-regexp] [-nocase] [-filter <arg>] [-quiet] [-verbose]
    [<patterns>...]
```

Returns

List of design objects.

Usage

Name	Description
`[-regexp]`	Patterns are full regular expressions
`[-nocase]`	Perform case-insensitive matching
`[-filter]`	Filter list with expression
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`[<patterns>]`	The patterns to match against Default: * Values: The default search pattern is the wildcard *, or .* when -regexp is specified.

Categories

[IPIntegrator](#)

Description

The command returns a list of example designs available in the current release of the Vivado Design Suite, or returns an error if it fails.

Example designs can be opened, or instantiated into the Vivado Design Suite, onto specific target hardware devices or boards.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_example_designs` based on property values on the designs. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the "example design" object, "NAME", "SUPPORTED_BOARDS" and "SUPPORTED_PARTS" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match example designs against the specified patterns. The default pattern is the wildcard '*' which gets a list of all supported example designs.

Examples

The following example returns reports the properties of each of the example designs in the current release:

```tcl
foreach x [get_example_designs] {
puts "********* Design $x **************"
report_property -all $x}
```

Note: The reported properties include the PARTS property which lists the compatible parts for the example design.

See Also

- instantiate_example_design

Send Feedback

get_files

Get a list of source files

Syntax

```
get_files [-regexp] [-nocase] [-filter <arg>] [-compile_order <arg>]
    [-used_in <arg>] [-references] [-all] [-of_objects <args>] [-quiet]
    [-verbose] [<patterns>]
```

Returns

List of file objects.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching (valid only when -regexp specified)
[-filter]	Filter list with expression
[-compile_order]	Get files by type and in compilation order (must use with -used_in)
[-used_in]	Get files used in a specific step (must use with -compile_order)
[-references]	Get files referenced in the provided objects (must use with -of_objects)
[-all]	Include all internal files.
[-of_objects]	Get 'file' objects of these types: 'file fileset ip reconfig_module'.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match file names against patterns Default: *

Categories

Object, Project

Description

Gets a list of files in the current project that match a specified search pattern. The default command gets a list of all files in the project.

Send Feedback

The `get_files` command returns a machine readable list of files in the project, in a design, or in a sub-design such as an IP core or block design. You can filter the results returned by `get_files` using one of the command arguments such as `-of_objects`, `-compile_order`, `-used_in`, and `-filter`.

The `-compile_order` and `-used_in` options must be used together to return a list of files from the sources or constraints filesets used in synthesis, simulation, or implementation, sorted according to the order of evaluation by the Vivado tools. The `-compile_order` and `-used_in` options use complex file ordering rules that can change based on header files, include files, or other language options. You can also filter files returned by `get_files` according to the USED_IN property, using the `-filter` option instead of the `-used_in` option.

You can use the `report_compile_order` command to generate a report of all files in the sources or constraints filesets, used in synthesis, simulation, AND implementation, sorted into different sections.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_files` based on property values on the files. You can find the properties on an object with the `report_property` or `list_property` commands. Any property/value pair can be used as a filter. In the case of the "file" object, "FILE_TYPE", and "IS_ENABLED" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <args> - (Optional) Get the files that are associated with the specified file, fileset, IP, or reconfig_module objects. The default is to search all filesets. When `-compile_order` and `-used_in` are specified, the `-of_objects` switch will only accept a single fileset, or a single sub-design such as an IP core, Block Design, or DSP design. A sub-design is also known as a composite file.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-compile_order [ sources | constraints ]` - (Optional) Returns the source design files, or the constraint files sorted according to the order of compilation by the Vivado Design Suite.

Note: This option must be used with the `-used_in` option. When specified, the `-of_objects` switch will only accept a single fileset or sub-design.

`-used_in` <arg> - (Optional) Accepts one of the enumerated values of the `USED_IN` property for files, and returns files matching the specified value. Valid values for this option include the following: synthesis, simulation, or implementation. Refer to the *Vivado Design Suite Properties Reference Guide (UG912)* for information on the USED_IN property and its supported values.

Note: This option must be used with the `-compile_order` option. When specified, the `-of_objects` switch will only accept a single fileset or sub-design.

`-all` - (Optional) Returns all of the files in the design, including internal files in support of IP and other objects.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

Send Feedback

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the set_msg_config command.

<patterns> - (Optional) Match files against the specified patterns. The default pattern is the wildcard '*' which gets all files in the project or of_objects. More than one pattern can be specified to find multiple files based on different search criteria.

Examples

The following example returns the VHDL files in the design that are used in simulation:

```
get_files -filter {FILE_TYPE == VHDL && USED_IN =~ simulation*}
```

This example returns the design source files that are used in simulation:

```
get_files -compile_order sources -used_in simulation
```

This example gets a list of files associated with the specified IP core composite file (ip.xci), from the sources_1 fileset that are used in synthesis:

```
get_files -compile_order sources -used_in synthesis -of [get_files ip.xci]
```

The following example gets a list of the files found in the sources_1 and constrs_1 filesets:

```
get_files -of [get_filesets {sources_1 constrs_1}]
```

Note: If there are no files matching the pattern you will get a warning.

See Also

- report_compile_order
- report_property

Send Feedback

get_filesets

Get a list of filesets in the current project

Syntax

```
get_filesets [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>]
    [-quiet] [-verbose] [<patterns>]
```

Returns

List of fileset objects.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching (valid only when -regexp specified)
[-filter]	Filter list with expression
[-of_objects]	Get 'fileset' objects of these types: 'reconfig_module'.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match fileset names against patterns Default: *

Categories

[Object](#), [Project](#)

Description

Gets a list of filesets in the current project that match a specified search pattern. The default command gets a list of all filesets in the project.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_filesets` based on property values on the filesets. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the fileset object, "DESIGN_MODE", and "FILESET_TYPE" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <args> - (Optional) Get the filesets that are associated with the specified reconfig_module objects. The default is to search all filesets.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Send Feedback

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match fileset names against the specified patterns. The default pattern is the wildcard '*' which gets all filesets. More than one pattern can be specified to find filesets based on multiple search criteria.

Examples

The following example returns the source files in the Source Set:

```
get_files -of_objects [get_filesets sources_1]
```

The following example makes `project_2` the active project, and then gets a list of filesets beginning with the letter s or the letter r:

```
current_project project_2
get_filesets s* r* -quiet
```

Note: If there are no filesets matching the pattern, such as r*, you will get a warning that no filesets matched the specified pattern. However, in the above example the use of `-quiet` will suppress that warning message.

The following example gets filesets beginning with the letter C, using a case insensitive regular expression:

```
get_filesets C.* -regexp -nocase
```

In the above example, constrs_1 and constrs_2 constraint sets would be returned if defined in the current project.

See Also

- get_files
- report_property

get_generated_clocks

Get a list of generated clocks in the current design

Syntax

```
get_generated_clocks [-regexp] [-nocase] [-filter <arg>]
    [-of_objects <args>] [-match_style <arg>] [-quiet] [-verbose]
    [<patterns>]
```

Returns

List of clocks.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching (valid only when -regexp specified)
[-filter]	Filter list with expression
[-of_objects]	Get generated clocks of these pins, ports or nets
[-match_style]	Style of pattern matching, valid values are ucf, sdc Default: sdc
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match generated clock names against patterns Default: *

Categories

[XDC](), [Object]()

Description

Gets a list of generated clocks in the current project that match a specified search pattern. The default command gets a list of all generated clocks in the project.

A generated clock can be added to the design using the `create_generated_clock` command, or can be automatically generated by the tool, at the output of an MMCM for instance.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_generated_clocks` based on property values on the generated clocks. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the generated_clock object, "DUTY_CYCLE" and "MASTER_CLOCK" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <args> - (Optional) Get the generated clocks connected to the specified port, pin, or net objects.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-match_style` - (Optional) Indicates that the search pattern matches UCF constraints or SDC constraints. The default is SDC.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match generated clocks against the specified patterns. The default pattern is the wildcard '*' which gets all generated clocks in the project.

Examples

The following example gets the names of all generated clocks in the current project:

```
get_generated_clocks
```

See Also

- create_generated_clock
- get_clocks
- report_property

get_gui_custom_command_args

Get custom command arguments

Syntax

```
get_gui_custom_command_args -command_name <arg> [-regexp] [-nocase]
    [-quiet] [-verbose] [<patterns>...]
```

Returns

List of custom command argument names.

Usage

Name	Description
-command_name	Unique name of the custom command whose arguments need to be displayed.
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching (valid only when -regexp specified)
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match the custom command argument names against patterns Default: *

Categories

GUIControl

Description

Returns the list of GUI custom command argument names for a particular GUI custom command.

You can use `get_gui_custom_commands` to determine the list of defined custom commands.

Arguments

`-command_name` - The name of the custom GUI command whose arguments should be returned.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match custom GUI command arguments against the specified patterns. The default pattern is the wildcard '*' which returns all arguments for the custom GUI command.

Examples

The following example returns a list of command arguments for the GUI custom command `abc`:

```
get_gui_custom_command_args -command_name abc
```

See Also

- create_gui_custom_command
- create_gui_custom_command_arg
- get_gui_custom_commands
- remove_gui_custom_command_args
- remove_gui_custom_commands

get_gui_custom_commands

Get custom commands

Syntax

```
get_gui_custom_commands [-regexp] [-nocase] [-quiet] [-verbose]
    [<patterns>...]
```

Returns

List of custom command names.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching (valid only when -regexp specified)
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match the custom command names against patterns Default: *

Categories

GUIControl

Description

Returns the list of GUI custom commands.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match custom GUI commands against the specified patterns. The default pattern is the wildcard '*' which returns all currently defined custom GUI commands.

Examples

The following example returns a list of all the GUI custom commands which has names starting with 'p':

```
get_gui_custom_commands p*
```

See Also

- [create_gui_custom_command](#)
- [create_gui_custom_command_arg](#)
- [get_gui_custom_command_args](#)
- [remove_gui_custom_command_args](#)
- [remove_gui_custom_commands](#)

get_hierarchy_separator

Get hierarchical separator character

Syntax

```
get_hierarchy_separator [-quiet] [-verbose]
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[SDC](), [XDC]()

Description

Gets the character currently used for separating levels of hierarchy in the design. You can set the hierarchy separator using the `set_hierarchy_separator` command.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example gets the currently defined hierarchy separator:

```
get_hierarchy_separator
```

See Also

- [set_hierarchy_separator]()

Send Feedback

get_highlighted_objects

Get highlighted objects

Syntax

```
get_highlighted_objects [-color_index <arg>] [-rgb <args>] [-color <arg>]
    [-quiet] [-verbose]
```

Returns

List of highlighted objects.

Usage

Name	Description
[-color_index]	Color index
[-rgb]	RGB color index list
[-color]	Valid values are red green blue magenta yellow cyan and orange
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Object, GUIControl

Description

Gets the highlighted objects in the current design open in the Vivado IDE. Objects can be highlighted with the `highlight_objects` command.

You can get all highlighted objects in the design, or specify highlighted objects by color, by color index, or by RGB value.

Note: This Tcl command works only when Vivado is run in GUI mode.

Arguments

`-color_index <arg>` - (Optional) Valid values are integers from 1 to 19. Specifies the color index used for highlighting the selected object or objects. The color index is defined by the **Colors → Highlight** category of the **Tools → Settings** command. Refer to the *Vivado Design Suite User Guide: Using the IDE (UG893)* for more information on setting themes.

`-rgb <args>` - (Optional) The color used for highlighting objects in the form of an RGB code specified as {R G B}. For instance, {255 255 0} specifies the color yellow.

`-color <arg>` - (Optional) The color used for highlighting the specified object or objects. Supported highlight colors are: red, green, blue, magenta, yellow, cyan, and orange.

Note: White is the color used to display selected objects with the `select_objects` command.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example gets all the highlighted objects in the current design:

```
get_highlighted_objects
```

The following example gets the object in the current design that are highlighted in the specified color:

```
get_highlighted_objects -color cyan
```

See Also

- get_marked_objects
- get_selected_objects
- highlight_objects
- mark_objects
- select_objects

get_hw_axi_txns

Get a list of hardware AXI transactions.

Syntax

```
get_hw_axi_txns [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
    [-quiet] [-verbose] [<patterns>]
```

Returns

Hw_axi_txns

Usage

Name	Description
[-of_objects]	Get 'hw_axi_txn' objects of these types: 'hw_axi'.
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match the 'hw_axi_txn' objects against patterns. Default: *

Categories

Hardware, Object

Description

Returns the read or write transactions for the specified JTAG to AXI Master core, hw_axi object.

The JTAG to AXI Master is a customizable IP core that works as an AXI Master to drive AXI transactions and drive AXI signals that are internal to the hardware device. This IP can be used in Vivado® IP integrator or can be instantiated in HDL in a Vivado project.

The JTAG-AXI core supports all memory-mapped AXI interfaces, except AXI4-Stream, and supports the AXI-Lite protocol. The AXI interface can be selected as a property of the core. The width of AXI data bus is customizable. This IP can drive any AXI4-Lite or Memory Mapped Slave directly, and can also be connected as the AXI Master to the interconnect. Run-time interaction with this core requires the use of the Vivado logic analyzer feature. Detailed documentation on the IP core can be found in the *LogiCORE IP JTAG to AXI Master Product Guide (PG174)*. A tutorial showing its use can be found in the *Vivado Design Suite Tutorial: Programming and Debugging (UG936)*.

Send Feedback

The JTAG to AXI Master core must be instantiated in the RTL code, from the Xilinx IP catalog. AXI transactions are defined as complete READ or WRITE transactions between the AXI master and various slaves.

This command returns a list of hw_axi_txn objects on the hw_device, or returns an error if it fails.

Arguments

`-of_objects <arg>` - (Optional) Return the AXI Master cores of the specified hardware devices. The devices must be specified as objects using the `get_hw_devices` or the `current_hw_device` commands.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter <args>` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_hw_axi_txns` based on property values on the AXI transactions. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the "hw_axi_txn" object, "NAME" and "TYPE" are two of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match hw_axi objects against the specified patterns. The default pattern is the wildcard '*' which gets a list of all hw_axis available on the current hardware device.

Example

The following example gets the ILA debug cores defined on the current hardware device:

```
get_hw_axi_txn -filter {TYPE==WRITE} [get_hw_axis]
```

See Also

- create_hw_axi_txn
- delete_hw_axi_txn
- get_hw_axis
- refresh_hw_axi
- reset_hw_axi

get_hw_axis

Get a list of hardware AXI objects.

Syntax

```
get_hw_axis [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
      [-quiet] [-verbose] [<patterns>]
```

Returns

Hw_axi

Usage

Name	Description
[-of_objects]	Get 'hw_axi' objects of these types: 'hw_device'.
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match the 'hw_axi' objects against patterns. Default: *

Categories

Hardware, Object

Description

Returns the JTAG to AXI Master core objects, or hw_axi objects, defined on the current hardware device.

The JTAG to AXI Master is a customizable IP core that works as an AXI Master to drive AXI transactions and drive AXI signals that are internal to the hardware device. This IP can be used in Vivado® IP integrator or can be instantiated from the Xilinx® IP catalog into the HDL of a Vivado project. Run-time interaction with this core requires the use of the Vivado logic analyzer feature.

The JTAG-AXI core supports all memory-mapped AXI interfaces, except AXI4-Stream, and supports the AXI-Lite protocol. The AXI interface can be defined as a property of the core. The width of the AXI data bus is customizable. This IP can drive any AXI4-Lite or Memory Mapped Slave directly, and can also be connected as the AXI Master to the interconnect. Detailed documentation on the IP core can be found in the *LogiCORE IP JTAG to AXI Master Product Guide (PG174).* A tutorial showing its use can be found in the *Vivado Design Suite Tutorial: Programming and Debugging (UG936).*

AXI cores can be found in the synthesized or implemented design netlist using the `get_debug_cores` command. This will return instances of the debug cores in the design that are related to the hw_axi objects, and other debug cores on the hw_device.

This command returns a list of AXI Master core objects, hw_axis, on the hw_device, or returns an error if it fails.

Arguments

`-of_objects <arg>` - (Optional) Return the AXI Master cores of the specified hardware devices. The devices must be specified as objects using the `get_hw_devices` or the `current_hw_device` commands.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter <args>` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_hw_axis` based on property values on the AXI Master cores. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the "hw_axi" object, "NAME" and "PROTOCOL" are two of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match hw_axi objects against the specified patterns. The default pattern is the wildcard '*' which gets a list of all hw_axis available on the current hardware device.

Example

The following example gets the ILA debug cores defined on the current hardware device:

```
get_hw_axis -of_objects [current_hw_device]
```

See Also

- connect_hw_server
- current_hw_device
- program_hw_devices
- set_property

get_hw_cfgmems

Get a list of hardware cfgmems.

Syntax

```
get_hw_cfgmems [-regexp] [-nocase] [-filter <arg>] [-quiet] [-verbose]
    [<patterns>]
```

Returns

Hardware cfgmems

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match the 'hw_cfgmem' objects against patterns. Default: *

Categories

[Hardware](), [Object]()

Description

Get a list of hardware configuration memory (hw_cfgmem) objects created for the current hw_device.

Xilinx FPGAs are configured by loading design-specific configuration data, in the form of a bitstream file, into the internal memory of the hw_device. The hw_cfgmem defines a flash memory device used for configuring and booting the Xilinx FPGA device. Once the hw_cfgmem object is created, and associated with the hw_device, the configuration memory can be programmed with the bitstream and other data using the `program_hw_cfgmem` command.

The hw_cfgmem object is associated with the specified hw_device object through the PROGRAM.HW_CFGMEM property on the device object. The `get_hw_cfgmems` command lets you work with the hw_cfgmem object.

This command returns a list of hw_cfgmem objects that match the specified search criteria, or returns an error if it fails.

Send Feedback

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_hw_cfgmems` based on property values on the hw_cfgmem objects. You can find the properties on an object with the `report_property` or `list_property` commands.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match hw_cfgmems against the specified patterns. The default pattern is the wildcard '*' which gets a list of all hw_cfgmems currently defined.

Example

The following example gets the hw_cfgmem objects associated with the current hw_device:

```
get_hw_cfgmems
```

See Also

- create_hw_cfgmem
- current_hw_device
- delete_hw_cfgmem
- get_hw_devices
- get_parts
- get_property
- program_hw_cfgmem
- set_property
- write_cfgmem

get_hw_ddrmcs

Get list of Versal integrated and soft DDRMC cores.

Syntax

```
get_hw_ddrmcs [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
    [-quiet] [-verbose] [<patterns>]
```

Returns

Integrated and soft DDRMC cores.

Usage

Name	Description
[-of_objects]	Get 'hw_ddrmc' objects of these types: 'hw_server hw_target hw_device'.
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match the 'hw_ddrmc' objects against patterns. Default: *

Categories

[Hardware](), [Object]()

Description

Xilinx Versal devices include integrated DDR Memory Controllers (DDRMC), as well as fabric based memory controllers. Refer to the *LogiCore IP Product Guide: Versal ACAP Network on Chip and Integrated Memory Controller* (PG313) for more information on the integrated controller core. Refer to *Versal ACAP Soft DDR4 Memory IP* (PG353) for more information on the fabric based memory controller. The memory controllers contain memory configuration and memory calibration data that can be accessed from the Xilinx Vivado Hardware Manager. The calibration and margin data are organized and presented in a graphical format for ease of analysis.

The `get_hw_ddrmc` command returns a list of memory cores on the current hardware device.

Arguments

`-of_objects <arg>` - (Optional) Return the Versal Memory controller (DDRMC) objects of the specified hw_server, hw_target, or hw_device objects.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter <args>` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_hw_ddrmcs` based on property values on the objects. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the hw_ddrmc object, "MEM_TYPE" and "BITS_PER_BYTE", are examples of couple of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Send Feedback

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match hw_ddrmc objects against the specified patterns. The default pattern is the wildcard `` `*` `` which gets a list of all hw_ddrmc available on the current hardware device.

Examples

The following example returns the HBM with the specified DISPLAY_NAME:

```
get_hw_hbms -filter {DISPLAY_NAME == HBM_2}
```

The following example returns the DDRMC with the specified DISPLAY_NAME:

```
get_hw_ddrmcs -filter {DISPLAY_NAME == DDRMC_1}
```

See Also

- current_hw_device
- current_hw_target
- refresh_hw_ddrmc
- report_hw_ddrmc

Send Feedback

get_hw_devices

Get a list of hardware devices.

Syntax

```
get_hw_devices [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
     [-quiet] [-verbose] [<patterns>]
```

Returns

Hardware devices

Usage

Name	Description
[-of_objects]	Get 'hw_device' objects of these types: 'hw_target'.
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match the 'hw_device' objects against patterns. Default: *

Categories

Hardware, Object

Description

Returns the hardware devices on the open target of a connected hardware server.

Each hardware target can have one or more Xilinx devices to program, or to use for debugging purposes. The current device is specified or returned by the `current_hw_device` command.

This command returns a list of available hardware device objects, or returns an error if it fails.

Arguments

`-of_objects <arg>` - (Optional) Return the hardware devices of the specified hardware targets. The targets must be specified as objects using the `get_hw_targets` or the `current_hw_target` commands.

Send Feedback

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_hw_devices` based on property values on the hardware device objects. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the "hw_device" object, "NAME" and "PART" are two of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Send Feedback

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match hw_devices against the specified patterns. The default pattern is the wildcard '*' which gets a list of all hw_devices available on the connected hardware server.

Example

The following example gets the list of available hardware device objects on the current hardware target:

```
get_hw_devices -of_objects [current_hw_target]
```

See Also

- connect_hw_server
- create_hw_device
- current_hw_device
- current_hw_target
- get_hw_targets
- open_hw_target

get_hw_hbms

Get list of hardware HBM cores.

Syntax

```
get_hw_hbms [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
    [-quiet] [-verbose] [<patterns>]
```

Returns

Hardware HBM cores.

Usage

Name	Description
[-of_objects]	Get 'hw_hbm' objects of these types: 'hw_server hw_target hw_device'.
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match the 'hw_hbm' objects against patterns. Default: *

Categories

[Hardware](), [Object]()

Description

Certain Xilinx® UltraScale+ devices include an integrated High-Bandwidth Memory (HBM) controller. Refer to the LogiCore IP Product Guide: AXI High Bandwidth Memory Controller (PG276) for more information on this core. The HBM controller and memory stacks contain both performance counters and temperature sensors that can be accessed through an HBM activity monitor (amon) from within the Xilinx Vivado hardware manager. The HBM activity monitor can be used to gain real-time access to performance monitoring and temperature sensors on the physical device. The Vivado hardware manager displays read, write, and overall throughput of data, as well as the device temperature. The activity monitor displays, captures, and exports data to a CSV file when it is running.

The `get_hw_hbms` command returns a list of HBM cores on the current hardware target.

Send Feedback

Arguments

`-of_objects <arg>` - (Optional) Return the High Bandwidth Memory controller (hbm) objects of the specified hw_server, hw_target, or hw_device objects.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter <args>` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_hw_hbms` based on property values on the objects. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the hw_hbm object, "DISPLAY_NAME" and "HBM_STACK_NUM", are a couple of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match hw_hbm objects against the specified patterns. The default pattern is the wildcard '*' which gets a list of all hw_hbms available on the current hardware device.

Examples

The following example returns the HBM with the specified DISPLAY_NAME:

```
get_hw_hbms -filter {DISPLAY_NAME == HBM_2}
```

This example returns all the HBM objects in the current device:

```
get_hw_hbms
```

See Also

- add_hw_hbm_pc
- commit_hw_hbm
- current_hw_device
- current_hw_target
- pause_hw_hbm_amon
- refresh_hw_hbm
- remove_hw_hbm_pc
- resume_hw_hbm_amon
- run_hw_hbm_amon
- stop_hw_hbm_amon

get_hw_ila_datas

Get a list of hardware ILA data objects.

Syntax

```
get_hw_ila_datas [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
      [-quiet] [-verbose] [<patterns>]
```

Returns

Hardware ILA data.

Usage

Name	Description
[-of_objects]	Get 'hw_ila_data' objects of these types: 'hw_ila hw_device'.
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match the 'hw_ila_data' objects against patterns. Default: *

Categories

Hardware, Object

Description

Returns the ILA data objects in the Vivado logic analyzer.

The `upload_hw_ila_data` command creates a hw_ila_data object in the process of moving the captured data from the ILA debug core, hw_ila, on the physical FPGA device, hw_device. The `read_hw_ila_data` command can also create a hw_ila_data object when reading an ILA data file from disk.

The hw_ila_data object can be viewed in the waveform viewer of the Vivado logic analyzer by using the `display_hw_ila_data` command, and can be written to disk using the `write_hw_ila_data` command.

This command returns a list of available hardware ILA data objects, or returns an error if it fails.

Send Feedback

Arguments

`-of_objects <arg>` - (Optional) Return the hw_ila_data objects of the specified hw_ila debug objects. The hw_ila cores must be specified as objects using the `get_hw_ilas` or the `current_hw_ila` commands.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter <args>` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_hw_ila_datas` based on property values on the hardware ILA data objects. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the "hw_ila_data" object, "NAME" and "TIMESTAMP" are two of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match hw_ila_data objects against the specified patterns. The default pattern is the wildcard '*' which gets a list of all hw_ila_data objects available in the Vivado logic analyzer.

Example

The following example gets the data objects for all the hardware ILA debug cores on the current device:

```
get_hw_ila_datas -of_objects [get_hw_ilas]
```

See Also

- current_hw_device
- current_hw_ila
- current_hw_ila_data
- display_hw_ila_data
- get_hw_devices
- get_hw_ilas
- read_hw_ila_data
- upload_hw_ila_data
- write_hw_ila_data

get_hw_ilas

Get a list of hardware ILA.

Syntax

```
get_hw_ilas [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
    [-quiet] [-verbose] [<patterns>]
```

Returns

Hardware ILAs

Usage

Name	Description
[-of_objects]	Get 'hw_ila' objects of these types: 'hw_device'.
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match the 'hw_ila' objects against patterns. Default: *

Categories

Hardware, Object

Description

Returns the ILA debug core objects defined on the current hardware device.

The Integrated Logic Analyzer (ILA) debug core lets you perform in-system debug of implemented designs, or design bitstreams, on a programmed Xilinx FPGA. The ILA core includes many advanced features of modern logic analyzers, including boolean trigger equations, and edge transition triggers. You can use the ILA core to probe specific signals of the design, to trigger on programmed hardware events, and capture data from the Xilinx FPGA in real-time. Refer to *LogiCORE IP Integrated Logic Analyzer (PG172)* for details of the ILA core.

Send Feedback

You can add ILA debug cores into the RTL source files of a design, or in the synthesized netlist using the `create_debug_core` command. Refer to the *Vivado Design Suite User Guide: Vivado Programming and Debugging (UG908)* for more information on adding debug cores and signal probes to the design. You can get the debug cores in the synthesized netlist by using the `get_debug_core` commands. However, though they are related, the design debug cores are separate from the hardware debug cores found in the Hardware Manager feature of the Vivado Design Suite.

Debug cores and probes are written to the probes file (`.ltx`) using the `write_debug_probes` command and associated with the hardware device, along with the bitstream file (`.bit`), using the PROBES.FILE and PROGRAM.FILE properties of the hw_device object. The hardware device is programmed with this information using the `program_hw_devices` command.

The steps to debug your design in hardware using an ILA debug core are:

1. Connect to the hardware server and target using `connect_hw_server` and `open_hw_target`.

2. Program the FPGA with the bitstream (`.bit`) and probes (`.ltx`) files using `program_hw_devices`.

3. Set up the ILA debug core trigger and capture controls using `set_property` to set properties of the ILA.

4. Arm the ILA debug core trigger using `run_hw_ila`.

5. View the captured data from the ILA debug core in the Waveform window of the Vivado logic analyzer feature using `display_hw_ila_data`.

6. Optionally use the VIO debug core to drive control signals and/or view design status signals. See `get_hw_vios` for more information.

7. Optionally use the JTAG-to-AXI Master debug core to run transactions to interact with various AXI slave cores in your design. See `get_hw_axis` for more information.

This command returns a list of ILA debug core objects on the device, or returns an error if it fails.

Arguments

`-of_objects <arg>` - (Optional) Return the ILA debug cores of the specified hardware devices. The devices must be specified as objects using the `get_hw_devices` or the `current_hw_device` commands.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_hw_ilas` based on property values on the ILA debug core objects. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the "hw_ila" object, "NAME" and "CONTROL.DATA_DEPTH" are two of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

Send Feedback

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match hw_ilas against the specified patterns. The default pattern is the wildcard '*' which gets a list of all hw_ilas available on the current hardware device.

Example

The following example gets the ILA debug cores defined on the current hardware device:

```
get_hw_ilas -of_objects [current_hw_device]
```

See Also

- connect_hw_server
- create_debug_core
- current_hw_device
- current_hw_ila
- display_hw_ila_data
- get_hw_axis
- get_hw_devices
- get_hw_ila_datas
- get_hw_vios
- open_hw_target
- program_hw_devices
- run_hw_ila
- set_property
- write_debug_probes

Send Feedback

get_hw_migs

Get list of hardware Migs cores.

Syntax

```
get_hw_migs [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
      [-quiet] [-verbose] [<patterns>]
```

Returns

Hardware migs cores.

Usage

Name	Description
[-of_objects]	Get 'hw_mig' objects of these types: 'hw_server hw_target hw_device'.
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match the 'hw_mig' objects against patterns. Default: *

Categories

Hardware, Object

Description

Returns the hw_mig objects on the current hardware device.

Memory IP included in the Xilinx IP catalog are used to generate memory controllers and interfaces for Xilinx® devices. Memory IP includes different IP cores from the Xilinx IP catalog depending on the device architecture and memory interface specified. Refer to *Zynq-7000 SoC and 7 Series Devices Memory Interface Solutions (UG586)*, or *UltraScale Architecture-Based FPGAs Memory Interface Solutions (PG150)*, for details of the available memory IP.

When added to a design, the memory IP needs to be implemented into the design. This occurs after the design is synthesized, during the design optimization phase of implementation, or `opt_design`, or can be done manually with the `implement_mig_cores` command.

A memory controller can be debug enabled when added into the design from the Xilinx IP catalog. In the Vivado logic analyzer, or the Vivado Lab Edition, memory controllers implemented into a design are associated with hw_mig objects, one hw_mig object per debug-enabled memory controller. The hw_mig object will have all the properties needed to get the calibration status and draw the per-bit eye margin views.

In the Vivado logic analyzer, or Vivado Lab Edition, the hw_mig object lets you verify the calibration and read window margins in your memory interface design. You can use the hardware manager GUI to check the calibration status, verify the read margin for both rising and falling edges of the clock or DQS. You can also use an ILA core to verify the data integrity for the read and write operations.

This command returns a list of hw_mig objects on the current hardware device, or returns an error if it fails.

Arguments

`-of_objects <arg>` - (Optional) Return the memory IP objects of the specified hw_server, hw_target, or hw_device objects.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter <args>` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_hw_migs` based on property values on the objects. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the hw_mig object, "DISPLAY_NAME", "BYTES", and "NIBBLES" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match hw_mig objects against the specified patterns. The default pattern is the wildcard '*' which gets a list of all hw_migs available on the current hardware device.

Example

The following example gets the memory IP defined on the current hardware target:

```
get_hw_migs -of_objects [current_hw_target]
```

See Also

- commit_hw_mig
- connect_hw_server
- current_hw_device
- current_hw_target

- implement_mig_cores
- refresh_hw_mig
- report_hw_mig
- set_property

Send Feedback

get_hw_pcies

Get list of soft PCIe core(s).

Syntax

```
get_hw_pcies [-of_objects <args>] [-regexp] [-nocase] [-patterns <arg>]
      [-filter <arg>] [-quiet] [-verbose]
```

Returns

PCIe core(s)

Usage

Name	Description
[-of_objects]	Get 'hw_pcie' objects of these types: 'hw_server hw_target hw_device'.
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-patterns]	Match the 'hw_pcie' objects against patterns. Default: *
[-filter]	Filter list with expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Hardware](), [Object]()

Description

Get all the Peripheral Component Interconnect Express (PCIe) debug core objects, hw_pcie's, defined on the current hardware device.

The customizable LogiCORE IP PCIe core for Xilinx ACAPs is designed for evaluating and monitoring the PCIe Link Training and Status State Machine (LTSSM) running on the Gigabit Transceivers (GTs). In the Vivado Hardware Manager, Versal PCIe soft cores implemented in the design, are represented as hw_pcie objects. You can use these PCIe debug cores to solve a range of debug and validation problems; from viewing the PCIe link info to the LTSSM state transition diagram.

This command returns a list of PCIe debug core objects on the device, or returns an error if it fails.

Send Feedback

Arguments

`-of_objects` - (Optional) Return the PCIe debug cores of the specified hw_server, hw_target or hw_device objects. The objects must be specified by an appropriate get_hw_* command.

Note: The `-of_objects` option requires objects to be specified using the get_* commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search.

`-filter <args>` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_hw_pcies` based on property values on the PCIe debug core objects. You can find the properties on an object with the report_property or list_property commands. In the case of the hw_pcie object, `NAME` is the only property that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||).

`<patterns>` - (Optional) Match hw_pcies against the specified patterns. The default pattern is the wildcard * which gets a list of all hw_pcies available on the current hardware device.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

Examples

The following example returns all the available PCIe cores

```
get_hw_pcies
```

The following example returns PCIe core with the specified DISPLAY_NAME

```
get_hw_pcies -filter {NAME == "PCIe_0"}
```

See Also

- reset_hw_pcie
- report_hw_pcie
- refresh_hw_pcie

get_hw_probes

Get a list of hardware probes.

Syntax

```
get_hw_probes [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
    [-quiet] [-verbose] [<patterns>]
```

Returns

Hardware probes

Usage

Name	Description
[-of_objects]	Get 'hw_probe' objects of these types: 'hw_interface hw_ila hw_vio'.
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match the 'hw_probe' objects against patterns. Default: *

Categories

Hardware, Object

Description

Returns the hw_probe objects in the Hardware Manager that are defined on signals in the design, or that are assigned to the specified ILA or VIO debug cores.

You can add ILA and VIO debug cores in the RTL source files of a design by customizing the core from the IP catalog, or add ILA debug cores into the synthesized netlist using the `create_debug_core` command.

Signals in the design can be probed to monitor signal values and track hardware events on the FPGA device. Debug probes can be added to ILA debug cores in the synthesized netlist design using the `create_debug_port` command, and connected to signals in the design using `connect_debug_port`. Probes can only be added to VIO debug cores when the IP core is customized, or re-customized, from the IP catalog, and signals connected to it in the RTL design. Refer to the *Vivado Design Suite User Guide: Vivado Programming and Debugging (UG908)* for more information on adding ILA and VIO debug cores and signal probes to the design.

Debug cores and probes are written to a probes file (.ltx) with `write_debug_probes`, and associated with the hardware device, along with the bitstream file (.bit), using the PROBES.FILE and PROGRAM.FILE properties of the hw_device object. The hardware device is programmed with this information using the `program_hw_devices` command.

This command returns a list of debug probe objects on the device, or returns an error if it fails.

Arguments

`-of_objects <arg>` - (Optional) Return the debug probes assigned to the specified ILA or VIO debug cores. The ILA or VIO cores must be specified as hardware objects using the `get_hw_ilas`, `current_hw_ila`, `get_hw_vios`, or `current_hw_vio` commands.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter <args>` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_hw_probes` based on property values on the debug probe objects. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the "hw_probes" object, "NAME" and "SIGNAL" are two of the properties that can be used to filter results.

Send Feedback

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match hw_probes against the specified patterns. The default pattern is the wildcard '*' which gets a list of all hw_probes assigned on the current device.

Example

The following example gets probes assigned to the current ILA debug core:

```
get_hw_probes -of_object [current_hw_ila]
```

See Also

- connect_debug_port
- create_debug_core
- create_debug_port
- current_hw_device

- current_hw_ila
- get_hw_devices
- get_hw_ilas
- get_hw_vios

Send Feedback

get_hw_servers

Get a list of hardware servers.

Syntax

```
get_hw_servers [-regexp] [-nocase] [-filter <arg>] [-quiet] [-verbose]
    [<patterns>]
```

Returns

Hardware servers

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match the 'hw_server' objects against patterns. Default: *

Categories

Hardware, Object

Description

> ⭐ **IMPORTANT!** *You must use the* `open_hw` *command to open the Hardware Manager in the Vivado Design Suite before using this command.*

This command returns hardware server objects that are connected to the current instance of the Vivado Design Suite through the use of the `connect_hw_server` command.

Hardware servers are instances of the Xilinx hardware server (hw_server) application running remotely, or on the local machine. The hardware server manages connections to a hardware target, for instance a hardware board containing a JTAG chain of one or more Xilinx devices to be used for programming and debugging your FPGA design.

This command returns a list of connected hardware server objects.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_hw_servers` based on property values on the hardware server objects. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the "hw_server" object, "HOST", "NAME" and "PORT" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match hw_servers against the specified patterns. The default pattern is the wildcard '*' which gets a list of all hw_servers currently connected.

Example

The following example returns a list of all hw_servers connected to the Vivado Design Suite:

```
get_hw_servers
```

See Also

- connect_hw_server
- current_hw_server
- disconnect_hw_server
- refresh_hw_server

get_hw_sio_commons

Get list of hardware SIO GT commons.

Syntax

```
get_hw_sio_commons [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
     [-quiet] [-verbose] [<patterns>]
```

Returns

Hardware SIO GT commons.

Usage

Name	Description
[-of_objects]	Get 'hw_sio_common' objects of these types: 'hw_server hw_target hw_device hw_sio_ibert hw_sio_gtgroup hw_sio_pll'.
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match the 'hw_sio_common' objects against patterns. Default: *

Categories

[Hardware](#), [Object](#)

Description

Returns the QPLL objects, hw_sio_commons, defined on the IBERT debug core on the current hardware device.

For each serial transceiver channel, there is a ring PLL called Channel PLL (CPLL). In Xilinx UltraScale and 7 series FPGAs, the high-speed transceivers have an additional common or shared PLL per quad, or Quad PLL (QPLL). This QPLL is a shared LC PLL to support high speed, high performance, and low power multi-lane applications.

On the device, the GTXE2_CHANNEL component has the serial transceiver and CPLL units and the GTXE2_COMMON has the QPLL unit.

This command returns a list of QPLL objects on the device as hw_sio_common objects, or returns an error if it fails.

Arguments

`-of_objects <arg>` - (Optional) Return the QPLL objects of the specified hw_server, hw_target, hw_device, hw_sio_ibert, or hw_sio_pll. The objects must be specified using the appropriate `get_hw_*` command.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter <args>` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_hw_sio_commons` based on property values on the hw_sio_common objects. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the "hw_sio_common" object, "NAME" and "DISPLAY_NAME" are two of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match hw_sio_commons against the specified patterns. The default pattern is the wildcard '*' which gets a list of all hw_sio_commons available on the current hardware device.

Example

The following example gets the ILA debug cores defined on the current hardware device:

```
get_hw_sio_commons -of [get_hw_sio_iberts]
```

See Also

- current_hw_device
- get_hw_devices
- get_hw_servers
- get_hw_sio_iberts
- get_hw_sio_plls
- get_hw_targets
- report_property

get_hw_sio_gtgroups

Get list of hardware SIO GT groups.

Syntax

```
get_hw_sio_gtgroups [-of_objects <args>] [-regexp] [-nocase]
    [-filter <arg>] [-quiet] [-verbose] [<patterns>]
```

Returns

Hardware SIO GT groups.

Usage

Name	Description
[-of_objects]	Get 'hw_sio_gtgroup' objects of these types: 'hw_server hw_target hw_device hw_sio_ibert hw_sio_common hw_sio_pll hw_sio_gt hw_sio_tx hw_sio_rx'.
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match the 'hw_sio_gtgroup' objects against patterns. Default: *

Categories

[Hardware](#), [Object](#)

Description

Returns the groups of GTs, hw_sio_gtgroups, as they relate to the quads that are in use on the IBERT debug core on the current hardware device.

GT groups, relate to the IO Banks on the hardware device, with the number of available banks and GT pins determined by the target device. On the Kintex-7 xc7k325 part, for example, there are four quads, each containing four differential GT pins, and two differential REFCLK pins. Each GT pin has its own transmitter, TX, and receiver, RX. Quads can also include one shared PLL per quad, or Quad PLL. The quads are defined on the IBERT debug core, and can be customized with a number of user settings when the IBERT is added into the RTL design. Refer to the *LogiCORE IP Integrated Bit Error Ratio Tester (IBERT) for 7 Series GTX Transceivers v3.0 (PG132)*or to *7 Series FPGAs GTX/GTH Transceivers User Guide (UG476)* or *UltraScale Architecture GTH Transceivers User Guide (UG576)* for more information.

Send Feedback

This command returns a list of GT group objects on the IBERT debug core, or returns an error if it fails.

Arguments

`-of_objects <arg>` - (Optional) Return the hw_sio_gtgroup objects of the specified hw_server, hw_target, hw_device, hw_sio_ibert, hw_sio_common, hw_sio_pll, hw_sio_gt, hw_sio_tx, or hw_sio_rx. The objects must be specified using the appropriate `get_hw_*` command.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter <args>` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_hw_sio_gtgroups` based on property values on the GT group objects. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the "hw_sio_gtgroup" object, "DISPLAY_NAME" and "GT_TYPE" are two of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match hw_sio_gtgroup objects against the specified patterns. The default pattern is the wildcard '*' which gets a list of all hw_sio_gtgroups available on the current hardware device.

Example

The following example gets the ILA debug cores defined on the current hardware device:

```
get_hw_sio_gtgroups -of [get_hw_sio_gts *MGT_X0Y9]
```

See Also

- current_hw_device
- get_hw_devices
- get_hw_servers
- get_hw_sio_commons
- get_hw_sio_gts
- get_hw_sio_iberts
- get_hw_sio_links
- get_hw_sio_plls
- get_hw_sio_rxs
- get_hw_sio_txs
- get_hw_targets
- report_property

get_hw_sio_gts

Get list of hardware SIO GTs.

Syntax

```
get_hw_sio_gts [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
     [-quiet] [-verbose] [<patterns>]
```

Returns

Hardware SIO GTs.

Usage

Name	Description
[-of_objects]	Get 'hw_sio_gt' objects of these types: 'hw_server hw_target hw_device hw_sio_ibert hw_sio_gtgroup hw_sio_pll hw_sio_tx hw_sio_rx hw_sio_link'.
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match the 'hw_sio_gt' objects against patterns. Default: *

Categories

[Hardware](), [Object]()

Description

Returns the Gigabit Transceiver objects (GTs), hw_sio_gt, that are in use on the IBERT debug core on the current hardware device.

The customizable LogiCORE™ IP Integrated Bit Error Ratio Tester (IBERT) core for Xilinx® FPGAs is designed for evaluating and monitoring the Gigabit Transceivers (GTs). The IBERT core enables in-system serial I/O validation and debug, letting you measure and optimize your high-speed serial I/O links in your FPGA-based system. Refer to the *LogiCORE IP Integrated Bit Error Ratio Tester (IBERT) for 7 Series GTX Transceivers v3.0 (PG132)*or to *7 Series FPGAs GTX/GTH Transceivers User Guide (UG476)* or *UltraScale Architecture GTH Transceivers User Guide (UG576)* for more information.

Using the IBERT debug core you can configure and tune the GT transmitters and receivers through the Dynamic Reconfiguration Port (DRP) port of the GTX transceiver. This lets you change property settings on the GTs, as well as registers that control the values on the ports.

This command returns a list of GT objects on the IBERT debug core, or returns an error if it fails.

Arguments

`-of_objects <arg>` - (Optional) Return the GT objects of the specified hw_server, hw_target, hw_device, hw_sio_ibert, hw_sio_pll, hw_sio_tx, hw_sio_rx, or hw_sio_link. The objects must be specified with an appropriate `get_hw_*` command.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter <args>` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_hw_sio_gts` based on property values on the GT objects. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the "hw_sio_gt" object, "NAME" and "GT_TYPE" are two of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match hw_sio_gt objects against the specified patterns. The default pattern is the wildcard '*' which gets a list of all hw_hw_sio_gts available on the current debug core.

Example

The following example gets the GTs assigned to the defined communication links on the IBERT debug core:

```
get_hw_sio_gts -of_objects [get_hw_sio_links]
```

See Also

- current_hw_device
- get_hw_devices
- get_hw_servers
- get_hw_sio_commons
- get_hw_sio_iberts
- get_hw_sio_links
- get_hw_sio_plls
- get_hw_sio_rxs
- get_hw_sio_txs
- get_hw_targets
- report_property

get_hw_sio_iberts

Get list of hardware SIO IBERT cores.

Syntax

```
get_hw_sio_iberts [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
      [-quiet] [-verbose] [<patterns>]
```

Returns

Hardware SIO IBERT cores.

Usage

Name	Description
[-of_objects]	Get 'hw_sio_ibert' objects of these types: 'hw_server hw_target hw_device hw_sio_gtgroup hw_sio_gt hw_sio_common hw_sio_pll hw_sio_tx hw_sio_rx hw_sio_link'.
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match the 'hw_sio_ibert' objects against patterns. Default: *

Categories

Hardware, Object

Description

Returns the Integrated Bit Error Ratio Tester (IBERT) debug core objects, hw_sio_ibert, defined on the current hardware device.

The customizable LogiCORE™ IP Integrated Bit Error Ratio Tester (IBERT) core for Xilinx® FPGAs is designed for evaluating and monitoring the Gigabit Transceivers (GTs). The IBERT core enables in-system serial I/O validation and debug, letting you measure and optimize your high-speed serial I/O links in your FPGA-based system. Refer to the *LogiCORE IP Integrated Bit Error Ratio Tester (IBERT) for 7 Series GTX Transceivers v3.0 (PG132)* for more information.

The IBERT debug core lets you configure and control the major features of GTs on the device, including:

• TX pre-emphasis and post-emphasis

Send Feedback

- TX differential swing

- RX equalization

- Decision Feedback Equalizer (DFE)

- Phase-Locked Loop (PLL) divider settings

You can use the IBERT core when you are interested in addressing a range of in-system debug and validation problems; from simple clocking and connectivity issues to complex margin analysis and channel optimization issues.

This command returns a list of IBERT debug core objects on the device, or returns an error if it fails.

Arguments

`-of_objects <arg>` - (Optional) Return the IBERT cores of the specified hw_server, hw_target, hw_device, hw_sio_gt, hw_sio_common, hw_sio_pll, hw_sio_tx, hw_sio_rx, or hw_sio_link objects. The objects must be specified by an appropriate `get_hw_*` command.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter <args>` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_hw_sio_iberts` based on property values on the IBERT debug core objects. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the "hw_sio_ibert" object, "NAME" and "CORE_REFRESH_RATE_MS" are two of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match hw_sio_ibert objects against the specified patterns. The default pattern is the wildcard '*' which gets a list of all hw_sio_ibert objects available on the current hardware device.

Example

The following example gets the ILA debug cores defined on the current hardware device:

```
get_hw_sio_iberts -of_objects [current_hw_device]
```

See Also

- current_hw_device
- get_hw_devices
- get_hw_servers

- get_hw_sio_commons
- get_hw_sio_gts
- get_hw_sio_links
- get_hw_sio_plls
- get_hw_sio_rxs
- get_hw_sio_txs
- get_hw_targets
- report_property

get_hw_sio_linkgroups

Get list of hardware SIO link groups.

Syntax

```
get_hw_sio_linkgroups [-of_objects <args>] [-regexp] [-nocase]
    [-filter <arg>] [-quiet] [-verbose] [<patterns>]
```

Returns

Hardware SIO link groups.

Usage

Name	Description
[-of_objects]	Get 'hw_sio_linkgroup' objects of these types: 'hw_sio_link'.
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match the 'hw_sio_linkgroup' objects against patterns. Default: *

Categories

[Hardware](), [Object]()

Description

Returns the defined groups, or associations of communication links between TX and RX objects on the GTs of the IBERT debug core defined on the current hardware device.

Vivado Serial I/O analyzer is a link-based analyzer. The links define the communication paths and protocols between transmitters and receivers of the GigaBit transceivers on the device. Link groups, or hw_sio_linkgroup objects, let you associate links into related groups, to collectively configure properties and run scans.

This command returns a list of linkgroups on the IBERT debug core, or returns an error if it fails.

Arguments

`-of_objects <arg>` - (Optional) Return the linkgroup objects of the specified hw_sio_links. The links must be specified as objects using the `get_hw_sio_links` command.

Send Feedback

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_hw_sio_linkgroups` based on property values on the link groups. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the "hw_sio_linkgroup" object, "NAME" and "DESCRIPTION" are two of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match hw_sio_linkgroup objects against the specified patterns. The default pattern is the wildcard '*' which gets a list of all hw_sio_linkgroups available on the current hardware device.

Example

The following example gets all of the link groups defined on the IBERT debug core on the current hardware device:

```
get_hw_sio_linkgroups
```

See Also

- create_hw_sio_link
- create_hw_sio_linkgroup
- current_hw_device
- get_hw_devices
- get_hw_servers
- get_hw_sio_iberts
- get_hw_sio_links
- get_hw_sio_rxs
- get_hw_sio_txs
- get_hw_targets
- remove_hw_sio_link
- remove_hw_sio_linkgroup
- report_property

get_hw_sio_links

Get list of hardware SIO links.

Syntax

```
get_hw_sio_links [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
     [-quiet] [-verbose] [<patterns>]
```

Returns

Hardware SIO links.

Usage

Name	Description
[-of_objects]	Get 'hw_sio_link' objects of these types: 'hw_server hw_target hw_device hw_sio_ibert hw_sio_gtgroup hw_sio_gt hw_sio_tx hw_sio_rx hw_sio_linkgroup'.
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match the 'hw_sio_link' objects against patterns. Default: *

Categories

[Hardware](), [Object]()

Description

Returns the communication links between TX and RX objects on the GTs of the IBERT debug core defined on the current hardware device.

Vivado Serial I/O analyzer is a link-based analyzer, which lets you link between any transmitter and receiver within the IBERT design. The links define the communication paths and protocols between transmitters and receivers of the GigaBit transceivers on the device. You can configure the links by using the `set_property` command to specify property values on the link object.

This command returns a list of link objects on the IBERT debug core, or returns an error if it fails.

Arguments

`-of_objects <arg>` - (Optional) Return the hw_sio_link objects of the specified hw_server, hw_target, hw_device, hw_sio_ibert, hw_sio_gt, hw_sio_tx, hw_sio_rx, or hw_sio_linkgroup. The objects must be specified using the appropriate `get_hw_*` command.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter <args>` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_hw_sio_links` based on property values on the link objects. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the "hw_sio_link" object, "DESCRIPTION", "TX_ENDPOINT", and "RX_ENDPOINT" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match hw_sio_links against the specified patterns. The default pattern is the wildcard '*' which gets a list of all hw_sio_link objects available on the current hardware device.

Example

The following example returns links based on the DESCRIPTION property that can be specified when the link is created:

```
get_hw_sio_links -filter {DESCRIPTION == "Link4" || DESCRIPTION == "Link2"}
```

See Also

- create_hw_sio_link
- create_hw_sio_linkgroup
- current_hw_device
- get_hw_devices
- get_hw_servers
- get_hw_sio_iberts
- get_hw_sio_linkgroups
- get_hw_sio_rxs
- get_hw_sio_txs
- get_hw_targets
- remove_hw_sio_link
- remove_hw_sio_linkgroup
- report_property

get_hw_sio_plls

Get list of hardware SIO PLLs.

Syntax

```
get_hw_sio_plls [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
    [-quiet] [-verbose] [<patterns>]
```

Returns

Hardware SIO PLLs.

Usage

Name	Description
[-of_objects]	Get 'hw_sio_pll' objects of these types: 'hw_server hw_target hw_device hw_sio_ibert hw_sio_gtgroup hw_sio_gt hw_sio_common'.
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match the 'hw_sio_pll' objects against patterns. Default: *

Categories

Hardware, Object

Description

Returns the PLL objects, hw_sio_pll, defined on the IBERT debug core on the current hardware device.

For each serial transceiver channel, there is a ring PLL called Channel PLL (CPLL). In Xilinx UltraScale and 7 series FPGAs, the GTX has an additional shared PLL per quad, or Quad PLL (QPLL). This QPLL is a shared LC PLL to support high speed, high performance, and low power multi-lane applications.

On the device, the GTXE2_CHANNEL component has the serial transceiver and CPLL units and the GTXE2_COMMON has the QPLL unit.

This command returns a list of all PLL objects, both CPLLs and QPLLs on the device, or returns an error if it fails.

Arguments

`-of_objects <arg>` - (Optional) Return the PLL objects of the specified hw_server, hw_target, hw_device, hw_sio_ibert, hw_sio_gt, or hw_sio_common objects. The objects must be specified using the appropriate `get_hw_*` command for the object.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter <args>` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_hw_sio_plls` based on property values on the PLLs. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the "hw_sio_pll" object, "NAME" and "PARENT" are two of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match hw_sio_pll objects against the specified patterns. The default pattern is the wildcard '*' which gets a list of all hw_sio_plls available on the current hardware device.

Example

The following example returns a list of the PLL objects on the IBERT debug core:

```
join [get_hw_sio_plls -of [get_hw_sio_iberts] ] \n
```

This example returns the PLL objects on the hw_sio_commons of the IBERT debug core:

```
join [get_hw_sio_plls -of [get_hw_sio_commons ]] \n
```

See Also

- current_hw_device
- get_hw_devices
- get_hw_servers
- get_hw_sio_commons
- get_hw_sio_gts
- get_hw_sio_iberts
- get_hw_sio_links
- get_hw_sio_rxs
- get_hw_sio_txs
- get_hw_targets
- report_property

get_hw_sio_rxs

Get list of hardware SIO RXs.

Syntax

```
get_hw_sio_rxs [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
     [-quiet] [-verbose] [<patterns>]
```

Returns

Hardware SIO RXs.

Usage

Name	Description
[-of_objects]	Get 'hw_sio_rx' objects of these types: 'hw_server hw_target hw_device hw_sio_ibert hw_sio_gtgroup hw_sio_gt hw_sio_link'.
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match the 'hw_sio_rx' objects against patterns. Default: *

Categories

Hardware, Object

Description

Returns the receiver objects, hw_sio_rx, of the Gigabit Transceivers (GTs) that are in use on the IBERT debug core on the current hardware device.

On the hardware device, each GT includes an independent receiver, which consists of a PCS and a PMA. High-speed serial data flows from traces on the board into the PMA of the GTX/GTH transceiver RX, into the PCS, and finally into the FPGA logic.

This command returns a list of receiver objects on the device, or returns an error if it fails.

Send Feedback

Arguments

`-of_objects <arg>` - (Optional) Return the receiver objects of the specified hw_server, hw_target, hw_device, hw_sio_ibert, hw_sio_gt, or hw_sio_link. The objects must be specified using the appropriate `get_hw_*` command.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter <args>` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_hw_sio_rxs` based on property values on the receivers. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the "hw_sio_rx" object, "DISPLAY_NAME" and "RX_DATA_WIDTH" are two of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match hw_sio_rx objects against the specified patterns. The default pattern is the wildcard '*' which gets a list of all hw_sio_rxs available on the current hardware device.

Example

The following example gets the receiver objects associated with the defined communication links on the IBERT debug core:

```
get_hw_sio_rxs -of [get_hw_sio_links]
```

See Also

- current_hw_device
- get_hw_devices
- get_hw_servers
- get_hw_sio_commons
- get_hw_sio_gts
- get_hw_sio_iberts
- get_hw_sio_links
- get_hw_sio_plls
- get_hw_sio_txs
- get_hw_targets
- report_property

get_hw_sio_scans

Get list of hardware SIO scans.

Syntax

```
get_hw_sio_scans [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
    [-quiet] [-verbose] [<patterns>]
```

Returns

Hardware SIO scans.

Usage

Name	Description
[-of_objects]	Get 'hw_sio_scan' objects of these types: 'hw_sio_rx hw_sio_link hw_sio_sweep'.
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match the 'hw_sio_scan' objects against patterns. Default: *

Categories

Hardware, Object

Description

Returns serial I/O analyzer scan objects for the IBERT debug core.

To analyze the margin of a given link, it is often helpful to run a scan of the link using the specialized Eye Scan hardware of Xilinx UltraScale devices or 7 Series FPGAs. The Vivado serial I/O analyzer feature lets you to create, run, and save link scans.

This command returns one or more hw_sio_scan objects, or returns an error if he command fails.

Arguments

`-of_objects <arg>` - (Optional) Return the hw_sio_scan objects of the specified hw_sio_rx, hw_sio_link, or hw_sio_sweep. The objects must be specified using the appropriate `get_hw_*` command.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_hw_sio_scans` based on property values on the scans. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the "hw_sio_scan" object, "NAME" and "DESCRIPTION" are two of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Send Feedback

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match hw_sio_scan objects against the specified patterns. The default pattern is the wildcard '*' which gets a list of all hw_sio_scans available on the IBERT debug core.

Example

The following example gets the serial I/O analyzer scans:

```
get_hw_sio_scans
```

See Also

- create_hw_sio_scan
- create_hw_sio_sweep
- current_hw_device
- get_hw_sio_sweeps
- remove_hw_sio_scan
- remove_hw_sio_sweep
- run_hw_sio_scan
- run_hw_sio_sweep
- stop_hw_sio_scan
- stop_hw_sio_sweep
- wait_on_hw_sio_scan
- wait_on_hw_sio_sweep
- write_hw_sio_scan
- write_hw_sio_sweep

get_hw_sio_sweeps

Get list of hardware SIO sweeps.

Syntax

```
get_hw_sio_sweeps [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
     [-quiet] [-verbose] [<patterns>]
```

Returns

Hardware SIO sweeps.

Usage

Name	Description
[-of_objects]	Get 'hw_sio_sweep' objects of these types: 'hw_sio_link hw_sio_scan'.
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match the 'hw_sio_sweep' objects against patterns. Default: *

Categories

[Hardware](), [Object]()

Description

Return the serial I/O analyzer link sweep objects defined on the IBERT debug core.

To analyze the margin of a given link, it is often helpful to run a scan of the link using the specialized features of Xilinx UltraScale devices or 7 series FPGAs. It can also be helpful to run multiple scans on a the link with different configuration settings for the GTs. This can help you determine which settings are best for your design. The Vivado serial I/O analyzer feature enables you to define, run, and save link sweeps, or collections of link scans run across a range of values.

This command returns a link sweep object that you can use with the `run_hw_sio_sweep` command to run analysis on the specified links, or GT receivers. You can also save the sweep scan to disk using the `write_hw_sio_sweep` command.

You can remove the created sweep object using `remove_hw_sio_sweep`.

This command returns one or more hw_sio_sweep objects, or returns an error if the command fails.

Arguments

`-of_objects <arg>` - (Optional) Return the serial I/O analyzer sweep object of the specified hw_sio_link or hw_sio_scan. The objects must be specified using the appropriate `get_hw_*` command.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter <args>` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_hw_sio_sweeps` based on property values on the hw_sio_sweep objects. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the "hw_sio_sweep" object, "NAME" and "DESCRIPTION" are two of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match hw_sio_sweep objects against the specified patterns. The default pattern is the wildcard '*' which gets a list of all hw_sio_sweeps available on the current hardware device.

Example

The following example gets the sweep scans on the IBERT debug core:

```
get_hw_sio_sweeps
```

See Also

- create_hw_sio_scan
- create_hw_sio_sweep
- current_hw_device
- get_hw_sio_scans
- remove_hw_sio_scan
- remove_hw_sio_sweep
- run_hw_sio_scan
- run_hw_sio_sweep
- stop_hw_sio_scan
- stop_hw_sio_sweep
- wait_on_hw_sio_scan
- wait_on_hw_sio_sweep

- write_hw_sio_scan
- write_hw_sio_sweep

get_hw_sio_txs

Get list of hardware SIO TXs.

Syntax

```
get_hw_sio_txs [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
     [-quiet] [-verbose] [<patterns>]
```

Returns

Hardware SIO TXs.

Usage

Name	Description
[-of_objects]	Get 'hw_sio_tx' objects of these types: 'hw_server hw_target hw_device hw_sio_ibert hw_sio_gtgroup hw_sio_gt hw_sio_link'.
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match the 'hw_sio_tx' objects against patterns. Default: *

Categories

Hardware, Object

Description

Returns the transmitter objects, hw_sio_tx, of the Gigabit Transceivers (GTs) that are in use on the IBERT debug core on the current hardware device.

On the hardware device, each GT includes an independent transmitter, which consists of a PCS and a PMA. Parallel data flows from the device logic into the FPGA TX interface, through the PCS and PMA, and then out the TX driver as high-speed serial data.

This command returns a list of transmitter objects on the device, or returns an error if it fails.

Send Feedback

Arguments

`-of_objects <arg>` - (Optional) Return the transmitter objects of the specified hw_server, hw_target, hw_device, hw_sio_ibert, hw_sio_gt, or hw_sio_link. The objects must be specified using the appropriate `get_hw_*` command.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter <args>` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_hw_sio_txs` based on property values on the transmitters. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the "hw_sio_tx" object, "DISPLAY_NAME" and "TXUSRCLK_FREQ" are two of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Send Feedback

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match hw_sio_tx objects against the specified patterns. The default pattern is the wildcard '*' which gets a list of all hw_sio_txs available on the current hardware device.

Example

The following example returns a list of the transmitters on the current device:

```
join [get_hw_sio_txs] \n
```

See Also

- current_hw_device
- get_hw_devices
- get_hw_servers
- get_hw_sio_commons
- get_hw_sio_gts
- get_hw_sio_iberts
- get_hw_sio_links
- get_hw_sio_plls
- get_hw_sio_rxs
- get_hw_targets
- report_property

Send Feedback

get_hw_softmcs

Get list of Versal soft memory controller cores.

Syntax

```
get_hw_softmcs [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
     [-quiet] [-verbose] [<patterns>]
```

Returns

Soft memory controller cores.

Usage

Name	Description
[-of_objects]	Get 'hw_softmc' objects of these types: 'hw_server hw_target hw_device'.
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match the 'hw_softmc' objects against patterns. Default: *

Categories

Hardware, Object

Description

Xilinx Versal devices can be instantiated with fabric based of memory controllers which are equipped with logic cores to extract debug data from the hardware. This form of debug cores is known as Versal SoftMC. Refer to the LogiCore IP Product Guide Versal ACAP Network on Chip and Integrated Memory Controller (PG313) for more information on the integrated controller core. Refer to Versal ACAP Soft DDR4 Memory IP (PG353) for more information on the fabric-based memory controller.

The memory controllers contain memory configuration and memory calibration data that can be accessed from the Xilinx Vivado Hardware Manager. The calibration and margin data are organized and presented in a graphical format for ease of analysis. The `get_hw_softmcs` command returns a list of soft memory cores on the current hardware device.

Send Feedback

Arguments

`-of_objects <arg>` - (Optional) Return the Versal Soft Memory controller (SoftMC) objects of the specified `hw_server, hw_target,` or `hw_device` objects.

The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins` rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search `<pattern>`.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and -filter expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add `".*"` to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command regexp is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of -regexp only.

`-filter <arg>` - (Optional) Filter the results list with the specified expression. The -filter argument filters the list of objects returned by `get_hw_softmcs` based on property values on the objects. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the `hw_softmc` object, `"MEM_TYPE"` and `"BITS_PER_BYTE"` are examples of couple of the properties that can be used to filter results.

> **IMPORTANT!** *The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard `"*"` character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.*

> **IMPORTANT!** *The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the `"*"` wildcard character, this will match a property with a defined value of `""`.*

For string comparison, the specific operators that can be used in filter expressions are "equal" (`==`), "not-equal" (`!=`), "match" (`=~`), and "not-match" (`!~`). Numeric comparison operators `<`, `>`, `<=`, and `>=` can also be used. Multiple filter expressions can be joined by AND and OR (`&&` and `||`). The following gets input pins that do NOT contain the `RESET` substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (bool) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

Send Feedback

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<patterns>` - (Optional) Match hw_softmc objects against the specified patterns. The default pattern is the wildcard `` `*` `` which gets a list of all hw_softmc available on the current hardware device.

Examples

The following example returns the SoftMC with the specified `DISPLAY_NAME`:

```
get_hw_softmcs -filter {DISPLAY_NAME == SoftMC_2}
```

See also

- current_hw_device
- current_hw_target
- refresh_hw_softmc
- report_hw_softmc

get_hw_sysmon_reg

Get the system monitor register value

Syntax

```
get_hw_sysmon_reg [-quiet] [-verbose] <hw_sysmon> <hexaddress>
```

Returns

Register value in Hex.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hw_sysmon>	hw_sysmon object
<hexaddress>	Hex address to read from

Categories

[Hardware](#)

Description

Returns the hex value of the system monitor register defined at the specified address of the specified hw_sysmon object.

The System Monitor (SYSMON) Analog-to-Digital Converter (ADC) is used to measure die temperature and voltage on the hw_device. The sysmon monitors the physical environment via on-chip temperature and supply sensors. The ADC can access up to 17 external analog input channels.

Data for the system monitor is stored in dedicated registers, called status and control registers, accessible through the hw_sysmon_reg object. Refer to the Register Interface in *UltraScale Architecture System Monitor User Guide (UG580)*, or *7 Series FPGAs and Zynq-7000 SoC XADC Dual 12-Bit 1 MSPS Analog-to-Digital Converter User Guide (UG480)* for more information on the addresses of specific system monitor registers.

Although the `get_hw_sysmon_reg` command lets you directly access the values stored in registers of the hw_sysmon object, the recommended procedure is to retrieve the values of registers as formatted properties of the hw_sysmon object. For example, the following code retrieves the TEMPERATURE on the system monitor as a formatted property of the hw_sysmon object rather than accessing the hex value of the sysmon register:

```
set opTemp [get_property TEMPERATURE [get_hw_sysmons]
```

The `get_property` command returns the TEMPERATURE as a formatted value in degrees Celsius, Fahrenheit, or Kelvin as determined by the TEMPERATURE_SCALE property on the hw_sysmon object.

> **TIP:** *You can also be sure the property value on the object is current by issuing the* `refresh_hw_sysmon` *command prior to* `get_property`.

The `get_hw_sysmon_reg` command returns the unformatted hex value of the hw_sysmon_reg object on the specified hw_sysmons at the specified address, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_sysmon> - (Required) Specify the hw_sysmon object to access the registers of. The hw_sysmon must be specified as an object as returned by the `get_hw_sysmons` command.

<hexaddress> - (Required) Specify the hex address of the register on the system monitor to return the value of. The address is specified as a hex value, and the value at the specified address is returned as a hex value.

Example

The following example gets the value of the hw_sysmon_reg at the specified address, 00, which relates to the XADC register storing the operating temperature of the current hw_device:

```
set opTemp [get_hw_sysmon_reg [lindex [get_hw_sysmons] 0 ] 00 ]
```

Note: The operating temperature is returned as a hex value.

See Also

- commit_hw_sysmon
- connect_hw_server
- current_hw_device
- get_hw_devices
- get_hw_sysmons
- open_hw_target
- refresh_hw_sysmon
- set_hw_sysmon_reg
- set_property

get_hw_sysmons

Get list of hardware SysMons.

Syntax

```
get_hw_sysmons [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
     [-quiet] [-verbose] [<patterns>]
```

Returns

Hardware sysmons

Usage

Name	Description
[-of_objects]	Get 'hw_sysmon' objects of these types: 'hw_server hw_target hw_device'.
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match the 'hw_sysmon' objects against patterns. Default: *

Categories

Hardware, Object

Description

Returns the Sysmon debug core objects defined on the current hardware device.

The System Monitor (SYSMON) Analog-to-Digital Converter (ADC) is used to measure die temperature and voltage on the hw_device. The Sysmon monitors the physical environment via on-chip temperature and supply sensors. The ADC provides a high-precision analog interface for a range of applications. The ADC can access up to 17 external analog input channels. Refer to *UltraScale Architecture System Monitor User Guide (UG580),* or *7 Series FPGAs and Zynq-7000 SoC XADC Dual 12-Bit 1 MSPS Analog-to-Digital Converter User Guide (UG480)* for more information on a specific device architecture.

The hw_sysmon data is stored in dedicated registers called status registers accessible through the hw_sysmon_reg object. The values of the system monitor registers can be returned by the `get_hw_sysmon_reg` command.

Every device that supports the system monitor will automatically have one or more hw_sysmon objects created when `refresh_hw_device` is called. When the hw_sysmon object is created, it is assigned a property for all the temperature and voltage registers, as well as the control registers. On the hw_sysmon object, the values assigned to the temperature and voltage registers are already translated to Celsius/Fahrenheit and Voltage.

Although you can use the `get_hw_sysmon_reg` command to access the hex values stored in registers of a system monitor, you can also retrieve values of certain registers as formatted properties of the hw_sysmon object. For example, the following code retrieves the TEMPERATURE property of the specified hw_sysmon object rather than directly accessing the hex value of the register:

```
set opTemp [get_property TEMPERATURE [get_hw_sysmons]
```

This command returns a list of hw_sysmon objects on the current or specified hw_device, or returns an error if it fails.

Arguments

`-of_objects <arg>` - (Optional) Return the hw_sysmon objects of the specified hw_server, hw_target, or hw_device. The objects must be specified using the appropriate `get_hw_*` command .

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter <args>` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_hw_sysmons` based on property values on the Sysmon cores. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the "hw_sysmon" object, "NAME", "VCCINT", and "VCCAUX" are some of the properties that can be used to filter results.

Send Feedback

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match hw_sysmons against the specified patterns. The default pattern is the wildcard '*' which gets a list of all hw_sysmons available on the current hardware device.

Example

The following example gets the hw_sysmon objects defined on the current hardware device:

```
get_hw_sysmons -of_objects [current_hw_device]
```

See Also

- connect_hw_server
- current_hw_device
- get_hw_devices
- get_hw_sysmon_reg

- open_hw_target
- program_hw_devices
- set_property

get_hw_targets

Get a list of hardware targets.

Syntax

```
get_hw_targets [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
    [-quiet] [-verbose] [<patterns>]
```

Returns

Hardware targets

Usage

Name	Description
[-of_objects]	Get 'hw_target' objects of these types: 'hw_server'.
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match the 'hw_target' objects against patterns. Default: *

Categories

Hardware, Object

Description

Returns the available hardware targets of the connected hardware servers.

The hardware target is a system board containing a JTAG chain of one or more Xilinx devices that you can program with a bitstream file, or use to debug your design. Connections between hardware targets on the system board and the Vivado Design Suite are managed by the Xilinx hardware server application, and the `connect_hw_server` command. Refer to *Vivado Design Suite User Guide: Programming and Debugging (UG908)* for a list of supported JTAG download cables and devices.

Use the `open_hw_target` command to open a connection to one of the available hardware targets. The open target is automatically defined as the current hardware target. Alternatively, you can define the current target with the `current_hw_target` command, and then open a connection to the current target. The Vivado Design Suite directs programming and debug commands to the open target through the hardware server connection.

Send Feedback

This command returns a list of available hardware targets through all connected hardware servers, or returns an error if it fails.

Arguments

`-of_objects <arg>` - (Optional) Return the hardware targets of the specified hardware server. The hardware server must be specified as a hw_server object using the `get_hw_servers` commands.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter <args>` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_hw_targets` based on property values on the targets. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the "hw_target" object, "NAME" and "IS_OPENED" are two of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match hw_targets against the specified patterns. The default pattern is the wildcard '*' which gets a list of all hw_targets available on the connected hardware server.

Example

The following example returns the available hardware targets on the currently connected hardware servers:

```
get_hw_targets
```

See Also

- connect_hw_server
- create_hw_target
- current_hw_target
- get_hw_servers
- open_hw_target

get_hw_vios

Get a list of hardware VIOs.

Syntax

```
get_hw_vios [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
    [-quiet] [-verbose] [<patterns>]
```

Returns

Hardware VIOs

Usage

Name	Description
[-of_objects]	Get 'hw_vio' objects of these types: 'hw_device'.
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match the 'hw_vio' objects against patterns. Default: *

Categories

[Hardware](), [Object]()

Description

Returns the Virtual Input/Output (VIO) debug core objects that are defined on the current hardware device, hw_device.

The Virtual Input/Output (VIO) debug core can both monitor and drive internal signals on a programmed Xilinx FPGA in real time. In the absence of physical access to the target hardware, you can use this debug feature to drive and monitor signals that are present on the physical device.

The VIO core has hardware probes, hw_probe objects, to monitor and drive specific signals on the design. Input probes monitor signals as inputs to the VIO core. Output probes drive signals to specified values from the VIO core. Values on the debug core are driven onto the signals at the probe using the `commit_hw_vio` command.

Send Feedback

The VIO debug core needs to be instantiated in the RTL code, therefore you need to know what nets you want monitor and drive prior to debugging the design. The IP catalog provides the VIO core under the Debug category. Detailed documentation on the VIO core can be found in the *LogiCORE IP Virtual Input/Output Product Guide (PG159)*.

This command returns a list of VIO debug core objects on the device, or returns an error if it fails.

Arguments

`-of_objects <arg>` - (Optional) Return the VIO debug cores of the specified hardware devices. The devices must be specified as objects using the `get_hw_devices` or the `current_hw_device` commands.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter <args>` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_hw_vios` based on property values on the VIO debug cores. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the "hw_vio" object, "NAME" and "INSTANCE_NAME" are two of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match hw_vios against the specified patterns. The default pattern is the wildcard '*' which gets a list of all hw_vios available on the current hardware device.

Example

The following example gets the ILA debug cores defined on the current hardware device:

```
get_hw_vios -of_objects [current_hw_device]
```

See Also

- commit_hw_vio
- connect_hw_server
- current_hw_device
- get_hw_probes
- program_hw_devices
- refresh_hw_vio
- reset_hw_vio_activity
- reset_hw_vio_outputs
- set_property

get_interfaces

Get a list of I/O port interfaces in the current design

Syntax

```
get_interfaces [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>]
    [-quiet] [-verbose] [<patterns>]
```

Returns

List of interface objects.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching (valid only when -regexp specified)
[-filter]	Filter list with expression
[-of_objects]	Get interfaces of these pins or nets
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match I/O port interfaces against patterns Default: *

Categories

[Object](#)

Description

Gets a list of IO interfaces in the current project that match a specified search pattern. The default command gets a list of all IO interfaces in the project.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_interfaces` based on property values on the interfaces. You can find the properties on an object with the `report_property` or `list_property` commands.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <args> - (Optional) One or more pins or nets to which the interfaces are assigned.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

Send Feedback

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - Match interfaces against the specified pattern. The default pattern is the wildcard '*' which gets a list of all interfaces in the project.

Examples

The following example gets a list of all interfaces in the project:

```
get_interfaces
```

See Also

- create_interface
- delete_interface

get_io_standards

Get a list of IO standards.

Syntax

```
get_io_standards [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>]
     [-quiet] [-verbose] [<patterns>]
```

Returns

IO standards

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-of_objects]	Get the IO standards of these bels, sites, package_pins, io_banks, ports.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match IO standards against patterns Default: *

Categories

[Object](#)

Description

Get a list of IOSTANDARDs available for use on the target device.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Send Feedback

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_io_standards` based on property values on the objects. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of IOSTANDARDs object, "NAME", "IS_DCI" and "IS_DIFFERENTIAL" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <arg> - (Optional) Get the IOSTANDARDs of package_pin, site, bel, io_bank, and port objects.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match IOSTANDARDs against the specified patterns. The default pattern is the wildcard '*' which gets a list of all IOSTANDARDs for use on the target part. More than one pattern can be specified to find multiple IOSTANDARDs based on different search criteria.

Note: You must enclose multiple search patterns in braces {} to present the list as a single element.

Examples

The following example gets a list of differential IOSTANDARDs available for use on the target device:

```
get_io_standards -filter {IS_DIFFERENTIAL}
```

Note: If there are no objects matching the pattern you will get a warning.

See Also

- list_property
- report_property

get_iobanks

Get a list of iobanks.

Syntax

```
get_iobanks [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>]
    [-quiet] [-verbose] [<patterns>]
```

Returns

Iobanks

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-of_objects]	Get the iobanks of these package_pins, ports, clock regions, slrs or sites.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match iobanks against patterns Default: *

Categories

[XDC](), [Object]()

Description

Gets a list of I/O Banks on the target device in the current project that match a specified search pattern. The default command gets a list of all I/O Banks on the target device.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Send Feedback

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_iobanks` based on property values on the I/O Banks. You can find the properties on an object with the `report_property` or `list_property` commands. Some of the properties that can be used with for an iobank object include "BANK_TYPE", "DCI_CASCADE", and "INTERNAL_VREF".

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <arg> - (Optional) Get a list of the I/O Banks connected to these objects. Valid object types are clock_regions, package_pins, ports, slrs and sites.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match I/O Banks against the specified pattern. The default pattern is the wildcard '*' which gets a list of all I/O Banks in the design.

Examples

The following example returns the I/O Bank of the specified package pin:

```
get_iobanks -of_objects [get_package_pins H4]
```

This example returns the GT Banks on the device:

```
get_iobanks -filter {BANK_TYPE == BT_MGT}
```

See Also

- get_package_pins
- get_pkgpin_bytegroups
- get_pkgpin_nibbles
- get_ports
- get_sites
- list_property
- place_ports
- report_property

get_ip_upgrade_results

Get a list of results for IP upgrades during the current project

Syntax

```
get_ip_upgrade_results [-srcset <arg>] [-quiet] [-verbose] [<objects>...]
```

Returns

List of IP upgrade results.

Usage

Name	Description
`[-srcset]`	(Optional) Specifies the source file set containing the upgraded IP Default: The current source fileset Values: Source set name
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`[<objects>]`	IP to be upgraded Values: IP instance(s) within the design, as returned by 'get_ips <instance name>' or 'get_bd_cells <cell name>'

Categories

Object, Project, IPFlow

Description

Returns the names of the upgrade_log files for the specified IPs.

This command is used by the Vivado IDE to populate the IP Status Report window with information from the upgrade_log file. You can use the command to quickly obtain the upgrade_log file name, and then use the appropriate file commands, to read or display the contents.

This command returns the upgrade_log file names of the specified IP objects, or returns an error if it fails.

Arguments

`-srcset <arg>` - (Optional) Specify an alternate source file set to examine for the specified IPs. This lets you change from the default, which is the `sources_1` fileset.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the set_msg_config command.

<objects> - (Optional) Specify the IP objects to report the upgrade results for. IP objects are returned by the get_ips command.

Example

The following example returns the upgrade_log filenames for all IPs in the current fileset:

```
get_ip_upgrade_results [get_ips]
```

The following example prints the upgrade logs for the upgraded IP to the Tcl console:

```
set ipDefs [get_ip_upgrade_results [get_ips]]
for {set x 0} {$x<[llength $ipDefs]} {incr x} {
    set ipRoot [file rootname [lindex $ipDefs $x]]
    puts "Upgrade Log for $ipRoot"
    set ipDir [get_property IP_DIR [get_ips $ipRoot]]
    set ipLog [lindex $ipDefs $x]
    set catLog [concat $ipDir/$ipLog]
    # Check for file existence, and read file contents
    if {[file isfile $catLog]} {
        # Open the file for read, save the File Handle
        set FH [open $catLog r]
        #puts "Open $FH"
        set content [read $FH]
        foreach line [split $content \n] {
            # The current line is saved inside $line variable
            puts $line
        }
        close $FH
        #puts "Close $FH"
    }
}
```

See Also

- current_fileset
- get_ips
- import_ip
- report_ip_status
- upgrade_ip

Send Feedback

get_ipdefs

Get a list of IP from the current IP Catalog

Syntax

```
get_ipdefs [-name] [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>]
     [-all] [-quiet] [-verbose] [<patterns>...]
```

Returns

List of Catalog IP objects.

Usage

Name	Description
[-name]	Match the pattern against IP display name instead of VLNV
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching
[-filter]	Filter list with expression
[-of_objects]	Get the IPDefs of the objects specified: IP inst or XCI file.
[-all]	Return hidden IP
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	The patterns to match against Default: * Values: The default search pattern is the wildcard *, or .* when -regexp is specified.

Categories

[Object](#), [IPFlow](#)

Description

Get a list of IP cores defined in the IP catalog of the current project, based on the specified search pattern. The default is to return all IP cores defined in the Vivado tools IP catalog.

By default, the search is based on the VLNV property of the IP cores in the catalog. You can specify the `-name` option to search on the display name of IP cores instead.

Arguments

`-name` - (Optional) Indicates that the specified `<pattern>` refers to the DISPLAY_NAME property of the IP instead of the VLNV property.

Send Feedback

> 💡 **TIP:** *In the case of multi-word display names, such as "Multiply Adder", you can only search for a single string, so you would need to search for *Multiply* or *Adder* to locate this IP core.*

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_ipdefs` based on property values on the objects. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the "ipdefs" object, "VLNV", "NAME" and "IS_AXI" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <args> - (Optional) Get the IP definitions for the specified IP instances or IP file (`.xci`) objects. Objects must be specified by the `get_ips` or `get_files` command.

`-all` - (Optional) Get the IP definitions from all defined IP catalogs and repositories. By default, the `get_ipdefs` command will return the IP cores from the standalone IP catalog. This option returns IP from both the standard Vivado tool IP catalog, and the IP integrator IP catalog.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match IP core definitions in the IP catalog against the specified search patterns. The default pattern is the wildcard '*' which gets a list of all IP cores in the catalog. More than one pattern can be specified to find multiple core definitions based on different search criteria.

Note: You must enclose multiple search patterns in braces, {}, or quotes, "", to present the list as a single element.

Examples

The following example returns a list of all IP cores with NAME property matching the specified pattern:

```
get_ipdefs -filter {NAME=~*agilent*}
```

> 💡 **TIP:** *The filter operator '=~' loosely matches the specified pattern.*

The following example returns a list of all AXI compliant IP cores:

```
get_ipdefs -filter {IS_AXI==1}
```

The following example returns IP from both the Vivado tools standard IP catalog and the IP integrator IP catalog:

```
get_ipdefs -all *axi_interconnect*
```

The following example filters the above results with the DESIGN_TOOL_CONTEXTS property to return only the IP integrator IP:

```
get_ipdefs -all *axi_interconnect* -filter {DESIGN_TOOL_CONTEXTS =~*IPI*}
```

> 💡 **TIP:** *By filtering on the DESIGN_TOOL_CONTEXTS property, you can identify IP from the IP integrator catalog instead of the Vivado tools standard catalog.*

In some cases, where multiple versions of an IP are returned, you can also filter on the UPGRADE_VERSIONS property to get as specific version or the latest IP version, as shown in the following example:

```
get_ipdefs -all *axi_interconnect* -filter {UPGRADE_VERSIONS == ""}
```

TIP: *The {UPGRADE_VERSIONS == ""} filter returns IP defs that have no upgrade, and so are the latest version.*

See Also

- create_ip
- generate_target
- get_ips
- import_ip
- report_property
- update_ip_catalog

Send Feedback

get_ips

Get a list of IPs in the current design

Syntax

```
get_ips [-regexp] [-nocase] [-all] [-filter <arg>] [-exclude_bd_ips]
    [-of_objects <args>] [-quiet] [-verbose] [<patterns>...]
```

Returns

List of IP objects.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching
[-all]	Include subcore IP in search
[-filter]	Filter list with expression
[-exclude_bd_ips]	Exclude all IP owned by a block design.
[-of_objects]	Get 'ip' objects of these types: 'ip file'.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match IP names against patterns Default: * Values: The default search pattern is the wildcard *, or .* when -regexp is specified.

Categories

Object, Project, IPFlow

Description

Get a list of IP cores in the current project based on the specified search pattern. The default command returns a list of all IPs in the project.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-all` - (Optional) Returns submodules of IP objects. Submodules are IP instances that are created as part of the generation of the parent IP core. By default, only parent IP objects used in the current project or design are returned, any IP cores used within those parent IP objects are not returned.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_ips` based on property values on the IP cores. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the "IP" object, "NAME" and "DELIVERED_TARGETS" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-exclude_bd_ips` - (Optional) Do not include IPs from the current design that belong to a block design object.

`-of_objects <arg>` - (Optional) Get a list of the IP cores of the specified IP or file objects.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match IP cores in the design against the specified search patterns. The default pattern is the wildcard '*' which gets a list of all IP cores in the project. More than one pattern can be specified to find multiple cores based on different search criteria.

Note: You must enclose multiple search patterns in braces, {}, or quotes, "", to present the list as a single element.

Examples

The following example returns a list of IP cores with names beginning with the string "EDK":

```
get_ips EDK*
```

See Also

- create_ip
- generate_target
- get_ipdefs
- import_ip
- update_ip_catalog

Send Feedback

get_lib_cells

Get a list of Library Cells

Syntax

```
get_lib_cells [-regexp] [-filter <arg>] [-nocase] [-include_unsupported]
     [-of_objects <args>] [-quiet] [-verbose] <patterns>
```

Returns

List of library cells.

Usage

Name	Description
[-regexp]	Patterns are regular expressions.
[-filter]	Filter list with expression.
[-nocase]	Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of -regexp only.
[-include_unsupported]	Include test-only library cells.
[-of_objects]	Get the library cells of the objects passed in here. Valid objects are cells or instances (ie, get_cells), cell pins (ie, get_pins) and library pins (ie, get_lib_pins).
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<patterns>	Match library cell names against patterns.

Categories

Object

Description

Get a list of cells in the library for the target part of the current design. Use this command to query and look for a specific library cell, or type of cell and to get the properties of the cells.

This command requires a hierarchical name which includes the library name as well as the cell name: lib_name/cell_name.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Send Feedback

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_lib_cells` based on property values on the cells. You can find the properties on an object with the `report_property` or `list_property` commands.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <arg> - (Optional) Get a list of library cells of specific cells, pins, or library pins.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Required) Match library cells against the specified patterns. The pattern must specify both the library name and the cell name.

Examples

The following example gets the number of the cells in the library for the target part in the current design, and then gets the number of AND type cells in that library:

```
llength [get_lib_cells [get_libs]/*]
795
llength [get_lib_cells [get_libs]/AND*]
18
```

The following example gets the library cell for the specified cell object:

```
get_lib_cells -of_objects [lindex [get_cells] 1]
```

See Also

- get_cells
- get_libs
- get_lib_pins
- list_property
- report_property

get_lib_pins

Get a list of Library Cell Pins

Syntax

```
get_lib_pins [-regexp] [-filter <arg>] [-nocase] [-of_objects <args>]
    [-quiet] [-verbose] <patterns>
```

Returns

List of library cell pins.

Usage

Name	Description
[-regexp]	Patterns are regular expressions
[-filter]	Filter list with expression
[-nocase]	Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of -regexp only.
[-of_objects]	Get the library cell pins of the objects passed in here. Valid objects are cells or instances (ie, get_cells), cell pins (ie, get_pins) and library cells (ie get_lib_cells).
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<patterns>	Match library cell pin names against patterns of the form <library cell pattern>/<library pin pattern>.

Categories

[Object](#)

Description

Gets a list of the pins on a specified cell of the cell library for the target part in the current design.

Note: This command requires a hierarchical name which includes the library name and cell name as well as the pins: lib_name/cell_name/pins.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Send Feedback

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_lib_pins` based on property values on the pins. You can find the properties on an object with the `report_property` or `list_property` commands.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <arg> - (Optional) Get a list of library cell pins of the specified cells, library cells, or pin objects.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Required) Match lib pins against the specified patterns. The pattern must specify the library name, cell name, and the pins.

Examples

The following example gets a list of all library cell pins:

```
get_lib_pins xt_virtex6/AND2/*
```

The following example gets a list of all pins, of all cells in the cell library for the target device:

```
get_lib_pins [get_libs]/*/*
```

See Also

- get_libs
- get_lib_cells
- list_property
- report_property

get_libs

Get a list of Libraries

Syntax

```
get_libs [-regexp] [-filter <arg>] [-nocase] [-quiet] [-verbose]
    [<patterns>]
```

Returns

List of libraries.

Usage

Name	Description
[-regexp]	Patterns are regular expressions
[-filter]	Filter list with expression
[-nocase]	Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of -regexp only.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match library names against patterns. Default: *

Categories

Object

Description

Gets the cell library for the target device in the current design. There is a library for each device family because there are primitives that may be available in one device family but not in others.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Send Feedback

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_libs` based on property values on the objects. You can find the properties on an object with the `report_property` or `list_property` commands.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match libraries against the specified patterns. The default pattern is the wildcard '*' which gets a list of all libraries in the project.

Examples

The following example gets the cell library for the target part:

```
get_libs
```

See Also

- get_lib_cells
- get_lib_pins
- list_property
- report_property

Send Feedback

get_macros

Get a list of macros in the current design

Syntax

```
get_macros [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>]
    [-quiet] [-verbose] [<patterns>]
```

Returns

List of macro objects.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching (valid only when -regexp specified)
[-filter]	Filter list with expression
[-of_objects]	Get macros of these cells
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match macro names against patterns Default: *

Categories

XDC, Object

Description

Gets a list of macros in the current design that match a specified search pattern. The default command returns a list of all macros in the design.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_macros` based on property values on the macros. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the "macro" object, "NAME", "ABSOLUTE_GRID" and "RLOCS" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <arg> - (Optional) Get the macros of the specified cell objects.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match macros against the specified patterns. The default pattern is the wildcard '*' which gets a list of all macros in the project. More than one pattern can be specified to find multiple macros based on different search criteria.

Note: You must enclose multiple search patterns in braces, {}, or quotes, "", to present the list as a single element.

Examples

The following example returns the properties currently assigned to the macro matching the specified search pattern:

```
report_property [get_macro *Macro1]
```

Note: If there are no macros matching the pattern you will get a warning.

See Also

- create_macro
- delete_macros
- list_property
- report_property
- update_macro

get_marked_objects

Get marked objects

Syntax

```
get_marked_objects [-rgb <args>] [-color <arg>] [-quiet] [-verbose]
```

Returns

List of marked objects.

Usage

Name	Description
`[-rgb]`	RGB color index list
`[-color]`	Valid values are red green blue magenta yellow cyan and orange
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution

Categories

Object, GUIControl

Description

Get the marked objects in the current design open in the Vivado IDE. Objects can be marked with the `mark_objects` command.

You can get all marked objects in the design, or specify marked objects by color, or by RGB value.

Note: This Tcl command works only when Vivado is run in GUI mode.

Arguments

`-rgb` **<args>** - (Optional) The color to use in the form of an RGB code specified as {R G B}. For instance, {255 255 0} specifies the color yellow, while {0 255 0} specifies green.

`-color <arg>` - (Optional) The color used for marking the specified object or objects. Supported colors are: red, green, blue, magenta, yellow, cyan, and orange.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Send Feedback

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example gets all the marked objects in the current design:

```
get_marked_objects
```

The following example gets the object in the current design that are marked in the specified color:

```
get_marked_objects -color yellow
```

See Also

- get_highlighted_objects
- highlight_objects
- mark_objects
- select_objects

get_methodology_checks

Get a list of Methodology rule check objects

Syntax

```
get_methodology_checks [-regexp] [-nocase] [-filter <arg>] [-abbrev <arg>]
    [-quiet] [-verbose] [<patterns>]
```

Returns

List of Methodology rule_check objects.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-abbrev]	Get the largest ID for this abbrev
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match the 'rule_check' objects against patterns. Default: *

Categories

[Methodology](), [Object]()

Description

Gets a list of the currently defined methodology checks. This list includes the factory defined methodology checks for process and timing.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Send Feedback

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of methodology checks returned by `get_methodology_checks` based on property values on the rule checks. You can find the properties on an object with the `report_property` or `list_property` commands.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-abbrev` <arg> - (Optional) Get the methodology checks associated with the specified name or abbreviation.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Send Feedback

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match methodology checks against the specified patterns. The default pattern is the wildcard '*' which gets all methodology checks.

Examples

The following command gets a list of all synthesis methodology checks:

```
get_methodology_checks SYNTH*
```

See Also

- get_methodology_violations
- list_property
- report_methodology
- report_property

get_methodology_violations

Get a list of Methodology violations from a previous report_methodology run

Syntax

```
get_methodology_violations [-name <arg>] [-regexp] [-filter <arg>]
     [-nocase] [-quiet] [-verbose] [<patterns>]
```

Returns

List of Methodology violation objects.

Usage

Name	Description
[-name]	Get the results with this name
[-regexp]	Patterns are full regular expressions
[-filter]	Filter list with expression
[-nocase]	Perform case-insensitive matching (valid only when -regexp specified)
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match methodology_violations against patterns Default: * Values: The default search pattern is the wildcard *, or .* when -regexp is specified.

Categories

Methodology, Object

Description

Gets a list of violation objects found in the design when the `report_methodology` command is run. The properties of individual violation objects can be queried using `report_property` or `list_property` commands for details of the violation.

Violation objects are associated with the cells, nets, pins, or ports in the current design, or sites on the current device. The design objects associated with a methodology violation object can be obtained using the `-of_objects` option of the appropriate `get_*` command, such as `get_cells`, or `get_nets` for instance.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Arguments

`-name <arg>` - (Optional) Get the violations associated with the named methodology result set. In this case the `report_methodology` command must also have been run with the `-name` option.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_methodology_violations` based on property values on the violations. You can find the properties on an object with the `report_property` or `list_property` commands.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match violations against the specified patterns. The default pattern is the wildcard '*' which gets all violations. More than one pattern can be specified to find multiple violations based on different search criteria.

Note: You must enclose multiple search patterns in braces, {}, or quotes, "", to present the list as a single element.

Examples

The following example reports the methodology violations found in the current design, then returns a list of all those violations:

```
report_methodology
get_methodology_violations
```

The following example generates list of violations in the named methodology report, and then gets the pins associated with any violations found:

```
report_methodology -name method_1
get_pins -of_objects [get_methodology_violations -name method_1]
```

See Also

- get_methodology_checks
- list_property
- report_methodology
- report_property

get_msg_config

Returns the current message count, limit, or the message configuration rules previously defined by the set_msg_config command.

Syntax

```
get_msg_config [-id <arg>] [-severity <arg>] [-rules] [-limit] [-count]
    [-quiet] [-verbose]
```

Usage

Name	Description
[-id]	The message id to match. Should be used in conjunction with -limit or -count Default: empty
[-severity]	The message severity to match. Should be used in conjunction with -limit or -count Default: empty
[-rules]	Show a table displaying all message control rules for the current project
[-limit]	Show the limit for the number of messages matching either -id or -severity that will be displayed
[-count]	Show the number of messages matching either -id or -severity that have been displayed
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Report](#)

Description

Returns the current message limit or count applied to a specified message ID or severity, or returns all message configuration rules defined in the current project. Message configuration rules are defined using the set_msg_config command.

When used with -count this command will display the total number of messages that have been generated with the matching message id, or for the specified severity.

> **IMPORTANT!** *The* get_msg_config *command reports the message count for the original CPU process from which Vivado was launched. Any sub-processes that the Vivado Design Suite launches, such as sub-processes used by the* launch_runs *command to launch synthesis and implementation runs, will not be reported in the message count. This can create confusion when the message count returned by* get_msg_config -count *is different from what is displayed in the Vivado IDE for instance, or different from what you expect. For this reason, the* -count *option is best used for non-project based designs.*

When used with `-limit` this command will display the current limit of messages with the matching message id, or for the specified severity.

When used with `-rules`, it will display a table of all message configuration rules currently in effect.

Note: You can only return the limit, the count, or the rules in a single `get_msg_config` command. An error is returned if more than one action is attempted.

Arguments

`-id <arg>` - (Optional) The message ID, which is included in all returned messages. For example, "Common 17-54" and "Netlist 29-28".

`-severity <value>` - (Optional) The severity of the message. There are five message severities:

- `ERROR` - An ERROR condition implies an issue has been encountered which will render design results unusable and cannot be resolved without user intervention.

- `{CRITICAL WARNING}` - A CRITICAL WARNING message indicates that certain input/ constraints will either not be applied or are outside the best practices for a FPGA family. User action is strongly recommended.

  Note: Since this is a two word value, it must be enclosed in {}.

- `WARNING` - A WARNING message indicates that design results may be sub-optimal because constraints or specifications may not be applied as intended. User action may be taken or may be reserved.

- `INFO` - An INFO message is the same as a STATUS message, but includes a severity and message ID tag. An INFO message includes a message ID to allow further investigation through answer records if needed.

- `STATUS` - A STATUS message communicates general status of the process and feedback to the user regarding design processing. A STATUS message does not include a message ID.

`-rules` - (Optional) Return the message configuration rules in the current project as defined by the `set_msg_config` command.

Note: When `-rule` is specified, all configuration rules for the current project are returned regardless of any message qualifier such as `-id` or `-severity`.

`-limit` - (Optional) Return the current limit of messages that match the message ID or the message severity.

`-count` - (Optional) Return the current count of messages that match the specified message ID or message severity.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example gets the current count of the specified INFO message:

```
get_msg_config -id "Common 17-81" -count
```

The following example returns the message configuration rules in the current project:

```
get_msg_config -rules
```

This example changes the severity of messages with the specified message ID, gets the current message configuration rules, and then shows two different command forms to reset the specific rule and restore the message:

```
set_msg_config -id "Common 17-361" -severity INFO -new_severity WARNING
get_msg_config -rules
--------------------
Message control rules currently in effect are:
Rule Name   Rule                                                    Current
Message Count
1  set_msg_config -ruleid {1} -id {Common 17-361} -severity {INFO} -
new_severity {WARNING} 0
--------------------
reset_msg_config -id "Common 17-361" -default_severity
reset_msg_config -ruleid {1}
```

> **TIP:** *In the preceding example, only one of the* `reset_msg_config` *commands is needed to reset the message.*

See Also

- reset_msg_config
- set_msg_config

get_net_delays

Get the routed or estimated delays in picoseconds on a net from the driver to each load pin.

Syntax

```
get_net_delays -of_objects <args> [-regexp] [-nocase] [-patterns <arg>]
    [-filter <arg>] [-to <args>] [-interconnect_only] [-quiet] [-verbose]
```

Returns

Net_delays

Usage

Name	Description
-of_objects	Get 'net_delay' objects of these types: 'net'.
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-patterns]	Match the 'net_delay' objects against patterns. Default: *
[-filter]	Filter list with expression
[-to]	Get the delay of the net to the given terminal(s) or port(s).
[-interconnect_only]	Include only interconnect delays. The default is to include the intra-site delay.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Timing, Netlist, Object

Description

Get delay objects for the specified nets in the current design, from the driver to each load pin, or to specified load pins, through specific pins.

The delay object contains properties defining the maximum and minimum delays for the fast and slow process corners. Use the `get_property` command to extract the property of interest from the delay object. Delay property values on the delay object are specified in picoseconds.

💡 **TIP:** *In most cases the Vivado tools return delay values specified in nanoseconds, but the delay object uses picoseconds.*

The values returned are calculated or estimated depending upon whether the net is routed. Delay values can include the actual delay of routed interconnect, or estimated net delays for unrouted nets. The net delay can also include delay through logic elements or device sites, or just include the interconnect delay.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

The `get_net_delays` command returns the delay objects for the specified nets, or returns an error if it fails.

Arguments

`-of_objects <arg>` - (Required) Get the net delays of the specified net objects. This option can be used to reduce the amount of data returned by the `get_net_delays` command.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_nets` in the case of the `get_net_delays` command.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-patterns <arg>` - (Optional) Match net_delays against the specified patterns. The default pattern is the wildcard '*' which gets a list of all net_delays in the project. More than one pattern can be specified to find multiple pins based on different search criteria.

Note: You must enclose multiple search patterns in braces, {}, or quotes, "", to present the list as a single element.

`-filter <args>` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_net_delays` based on property values on the delay objects. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the delay object, "NET", "FAST_MAX" and "FAST_MIN" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-to` <args> - (Optional) Specifies the endpoints of nets for delay calculation. Pin, port, and pip objects can be specified as endpoints.

`-interconnect_only` - (Optional) Include only interconnect delays to determine the delay on the net due to routing. The default is to also include the intra-site delay into the delay calculation.

> 💡 **TIP:** *When using `get_net_delays` to define delay limits for the `route_design` command `-min_delay` or `-max_delay` options, you should use the `-interconnect_only` option to ensure the specified net delay includes just the interconnect.*

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example gets the interconnect delay values for the specified net, and returns it in the form of a delay object:

```
report_property -all [lindex [get_net_delays -interconnect_only \
-of_objects [get_nets control_reg[*]]] 16 ]
```

TIP: *The FAST_MAX, FAST_MIN, SLOW_MAX, and SLOW_MIN properties on the delay object are reported in picoseconds.*

See Also

- get_pins
- get_pips
- get_ports
- get_property
- list_property
- report_property

get_nets

Get a list of nets in the current design

Syntax

```
get_nets [-hsc <arg>] [-hierarchical] [-regexp] [-nocase] [-filter <arg>]
    [-of_objects <args>] [-match_style <arg>]
    [-top_net_of_hierarchical_group] [-segments] [-boundary_type <arg>]
    [-quiet] [-verbose] [<patterns>]
```

Returns

List of net objects.

Usage

Name	Description
[-hsc]	Hierarchy separator Default: /
[-hierarchical]	Search level-by-level in current instance
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching (valid only when -regexp specified)
[-filter]	Filter list with expression
[-of_objects]	Get nets of these pins, ports, cells, timing paths or clocks, drc violations
[-match_style]	Style of pattern matching, valid values are ucf, sdc Default: sdc
[-top_net_of_hierarchical_group]	Return net segment(s) which belong(s) to the high level of a hierarchical net
[-segments]	Return all segments of a net across the hierarchy
[-boundary_type]	Return net segment connected to a hierarchical pin which resides at the same level as the pin (upper) or at the level below (lower), or both. Valid values are : upper, lower, both Default: upper
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match net names against patterns Default: *

Categories

SDC, XDC, Object

Send Feedback

Description

Gets a list of nets in the current design that match a specified search pattern. The default command gets a list of all nets in the current_instance of the open design, as specified by the `current_instance` command.

You can use the `-hierarchical` option to extract nets from the hierarchy of the current design.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Arguments

`-hsc <arg>` - (Optional) The default hierarchy separator is '/'. Use this argument to specify a different hierarchy separator.

`-hierarchical` - (Optional) Get nets from all levels of the design hierarchy starting at the current instance. Without this argument, the command will only get nets from the current instance, as set by the `current_instance` command. When using `-hierarchical`, the search pattern should not contain a hierarchy separator because the search pattern is applied at each level of the hierarchy, not to the full hierarchical net name. For instance, searching for U1/* searches each level of the hierarchy for nets with U1/ in the name. This may not return the intended results. See `get_cells` for examples of `-hierarchical` searches.

Note: When used with `-regexpr`, the specified search string is matched against the full hierarchical name, and the U1/* search pattern will work as intended.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter <args>` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_nets` based on property values on the nets. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the nets object, "PARENT", "TYPE" and "MARK_DEBUG" are some of the properties that can be used to filter results.

Send Feedback

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <arg>` - (Optional) Get a list of the nets connected to the specified cells, pins, ports, or clocks; or nets associated with specified DRC violation objects.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-match_style [sdc | ucf]` - (Optional) Indicates that the search pattern matches UCF constraints or SDC constraints. The default is SDC.

`-top_net_of_hierarchical_group` - (Optional) Returns the top-level net segment of a hierarchical net, or nets, or the top-level net segments of all nets. Use this option with `-segment` to return the top-level net segment from all the segments of a hierarchical net.

`-segments` - (Optional) Get all the segments of a hierarchical net, across all levels of the hierarchy. This differs from the `-hierarchical` argument in that it returns all segments of the specified net, rather than just the specified net.

IMPORTANT! *The `-segments` option is applied after the `-filter` option has eliminated nets that do not match the filter pattern. Because of this, you may expect to see net segments returned that have already been filtered out of the returned results. To change this behavior, you can use the `filter` command rather than the `-filter` option to apply the `-segments` option to the `get_nets` command, and then filter the segments returned. See the Examples for more information.*

Send Feedback

`-boundary_type [ upper | lower | both ]` - (Optional) Gets the net segment at the level (upper) of a specified hierarchical pin, at the level below (lower) the pin or port, or both the level of and the level below. Valid values are upper, lower, or both. The default value is upper.

Note: This argument must be used with the `-of_objects` argument to specify the hierarchical pin.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match nets against the specified patterns. The default pattern is the wildcard '*' which returns a list of all nets in the project. More than one pattern can be specified to find multiple nets based on different search criteria.

Note: You must enclose multiple search patterns in braces, {}, or quotes, "", to present the list as a single element.

Examples

The following example runs the `report_drc` command on the current design, returns the list of violations in the specified DRC report, and then returns any nets associated with the driverless net rule (NDRV):

```
report_drc -name drc_1
get_drc_violations -name drc_1
get_nets -of_objects [get_drc_violations -name drc_1 NDRV*]
```

The following example returns a list of nets that have been marked for debug with the connect_debug_port command:

```
get_nets -hier -filter {MARK_DEBUG==1}
```

This example returns the net attached to the specified hierarchical pin object, then returns the net segments attached to the pin object, then returns the top-level net segment attached to the pin object:

```
get_nets \
   -of [get_pins cpuEngine/or1200_cpu/or1200_sprs/esr_reg[9]_i_3/I0]
get_nets -segments \
   -of [get_pins cpuEngine/or1200_cpu/or1200_sprs/esr_reg[9]_i_3/I0]
get_nets -top -segments \
   -of [get_pins cpuEngine/or1200_cpu/or1200_sprs/esr_reg[9]_i_3/I0]
```

In the following example, the first command applies the `-filter` to find nets that have the IS_INTERNAL property, and then `-segment` is applied to return the segments of those nets. This command returns 0 net segments (and a warning). The second command, returns the segments of all nets, and filters the results to find the segments that have the IS_INTERNAL property, of which there are 448:

```
llength [get_nets -segments -filter {IS_INTERNAL}]
WARNING: [Vivado 12-1023] No nets matched for command 'get_nets -segments
-filter IS_INTERNAL'.
0
llength [filter [get_nets -segments] {IS_INTERNAL}]
448
```

See Also

- connect_debug_port
- current_instance
- get_cells
- get_clocks
- get_drc_violations
- get_pins
- get_ports
- list_property
- report_drc
- report_property

Send Feedback

get_nodes

Get a list of nodes in the device.

Syntax

```
get_nodes [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
    [-uphill] [-downhill] [-flyover] [-from <args>] [-to <args>] [-quiet]
    [-verbose] [<patterns>]
```

Returns

Nodes

Usage

Name	Description
[-of_objects]	Get 'node' objects of these types: 'net tile node bel_pin site_pin wire pip speed_model'.
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-uphill]	Get the nodes uphill (driver) from the site_pin, pip, node or tile(s) provided in the -of_objects.
[-downhill]	Get the nodes downhill (loads) from the site_pin, pip, node or tile(s) provided in the -of_objects.
[-flyover]	Get the nodes that fly over the given tile(s).
[-from]	-from <pip/site pin> Return the nodes beginning at this pip or site pin. May be used in combination with uphill. Default is downhill. -all is implied.
[-to]	-to <pip/site pin> Return the nodes ending at this wire or site pin. May be used in combination with uphill. Default is downhill. -all is implied.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match the 'node' objects against patterns. Default: *

Categories

[Object](#), [XDC](#)

Description

Returns a list of nodes on the device that match a specified search pattern in an open design.

The default command gets a list of all nodes on the device.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Arguments

`-of_objects` <args> - (Optional) Return the nodes of the specified nets, tiles, nodes, bel_pins, site_pins, wires, speed_model or pip objects.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_nodes` based on property values on the nodes. You can find the properties on an object with the `report_property` or `list_property` commands. Any property/value pair can be used as a filter. In the case of the node object, "IS_INPUT_PIN", "IS_BEL_PIN" and "NUM_WIRES" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

Send Feedback

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-uphill` - (Optional) Return the nodes uphill of objects specified by the `-of_objects` option. Uphill nodes precede the specified object in the logic network.

`-downhill` - (Optional) Return the nodes downhill of objects specified by the `-of_objects` option. Downhill nodes follow the specified object in the logic network.

`-flyover` - (Optional) Return the nodes that pass through (or flyover) the specified tiles.

`-from` <args> - (Optional) Defines the starting points of the nodes to get from site_pin or pip objects. This option can be combined with `-uphill`. The default is to return nodes downhill of a start point. By default, all nodes will be returned.

`-to` <args> - (Optional) Defines the ending points of the nodes to get from site_pin or pip objects. This option can be combined with `-uphill`. The default is to return nodes downhill of a start point to the specified end point. By default, all nodes will be returned.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Return nodes matching the specified search patterns. The default pattern is the wildcard '*' which gets a list of all nodes on the device. More than one search pattern can be specified to find nodes based on different search criteria.

Note: You must enclose multiple search patterns in braces, {}, or quotes, "", to present the list as a single element.

Send Feedback

Examples

The following example returns the nodes associated with the specified tile:

```
get_nodes -of_objects [get_tiles CLBLM_R_X11Y158]
```

The following example returns the nodes downhill from the specified node:

```
get_nodes -downhill -of_objects [get_nodes LIOB33_SING_X0Y199/IOB_PADOUT0]
```

See Also

- get_site_pins
- get_tiles
- get_wires
- list_property
- report_property

get_objects

Get a list of HDL objects in one or more HDL scopes as per the specified pattern

Syntax

```
get_objects [-filter <arg>] [-r] [-local] [-regexp] [-nocase] [-quiet]
    [-verbose] [<patterns>...]
```

Returns

Returns all the objects found given the specified pattern.

Usage

Name	Description
[-filter]	filters <patterns> according to the specified property-matching expressions
[-r]	Searches recursively for objects
[-local]	Searches objects in the subprogram frame selected for the current scope
[-regexp]	Search using regular expressions, search design objects from which to create wave objects by design object name. The application supplying the design objects determines how the match is to be performed. Items must be strings.
[-nocase]	Perform a case insensitive match (only used with regexp)
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Patterns to search for. Default is * where all HDL objects are returned

Categories

[Simulation](#)

Description

Returns a list of HDL objects matching the specified search pattern in one or more HDL scopes.

HDL objects include HDL signals, variables, or constants as defined in the Verilog or VHDL test bench and source files. An HDL signal includes Verilog wire or reg entities, and VHDL signals. Examples of HDL variables include Verilog real, realtime, time, and event. HDL constants include Verilog parameters and localparams, and VHDL generic and constants.

The HDL scope, or scope, is defined by a declarative region in the HDL code such as a module, function, task, process, or begin-end blocks in Verilog. VHDL scopes include entity/architecture definitions, block, function, procedure, and process blocks.

Arguments

`-r` - (Optional) Apply the command to the current scope, and all sub-scopes of the current scope.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_objects` based on property values on the objects. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the HDL object, "NAME", "SCOPE" and "TYPE" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match HDL objects against the specified patterns. The default pattern is the wildcard '*' which returns all the children in the current scope. The search pattern can be defined in two ways:

- <patterns> - Specifies only the search pattern for the objects to get. This method returns all objects in the current scope (and any sub-scopes when `-recursive` is used).

- `<scope>/<pattern>` - Specifies the scope of interest, relative to the current scope, and the pattern for objects to locate. In this case, the specified `<scope>`, and any sub-scopes of it if `-recursive` is used, are identified starting from the current scope. Then all objects matching the search `<pattern>` are identified and returned.

Examples

The following example specifies the current_scope, then gets all HDL objects in that scope:

```
current_scope ./cpuEngine
get_objects
```

The following example returns the count of all objects in the current scope, and then returns the count of all objects in the current scope, and all sub-scopes of it:

```
llength [get_objects]
   182
llength [get_objects -recursive ]
   2182
```

The following example specifies the `<scope>/<pattern>` search pattern as discussed above. Notice that the cpuEngine scope and various sub-scopes of it are identified, then objects matching the `cl*` pattern in those scopes are returned:

```
get_objects -recursive -filter {type == internal_signal} cpuEngine/cl*
   /top/cpuEngine/clk_i
   /top/cpuEngine/iwb_biu/clk
   /top/cpuEngine/iwb_biu/clmode
   /top/cpuEngine/or1200_cpu/clk
   ...
   /top/cpuEngine/or1200_immu_top/or1200_immu_tlb/itlb_mr_ram/clk
```

Search the current scope, and all sub-scopes, for any internal signals whose names start with `cl` or `ma`:

```
get_objects -recursive -filter {type == internal_signal} ma* cl*
```

Send Feedback

See Also

- current_scope
- list_property
- report_objects
- report_property

get_package_pins

Get a list of package pins

Syntax

```
get_package_pins [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>]
    [-quiet] [-verbose] [<patterns>]
```

Returns

List of package pin objects.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-of_objects]	Get the list of package pin objects of these sites, bels, iobanks, pkgpin_bytegroups, pkgpin_nibbles, or ports.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match list of package pin objects against patterns Default: *

Categories

[XDC](), [Object]()

Description

Gets a list of the pins on the selected package for the target device. The default command gets a list of all pins on the package.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Send Feedback

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_package_pins` based on property values on the pins. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the package pin object, "IS_CLK_CAPABLE", "IS_VREF" and "IS_GLOBAL_CLK" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <arg>` - (Optional) Get the package pins connected to the specified objects. Valid objects include sites, bels, iobanks, pkgpin_bytegroups, pkgpin_nibbles, or ports.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match pins against the specified patterns. The default pattern is the wildcard '*' which returns all pins on the package. More than one pattern can be specified to find multiple pins based on different search criteria.

Examples

The following example gets a list of the package pins associated with the specified bytegroup of an UltraScale device:

```
get_package_pins -of [get_pkgpin_bytegroups BANK44_BYTE0]
```

The following example gets the number of clock capable (CC) pins on the package:

```
llength [get_package_pins -filter {IS_CLK_CAPABLE==1}]
```

Note: If there are no pins matching the pattern you will get a warning.

See Also

- get_pkgpin_bytegroups
- get_pkgpin_nibbles
- get_ports
- get_sites
- list_property
- place_ports
- report_property

Send Feedback

get_param

Get a parameter value

Syntax

```
get_param [-quiet] [-verbose] <name>
```

Returns

Parameter value

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<name>	Parameter name

Categories

[PropertyAndParameter](#)

Description

This command gets the currently defined value for a specified tool parameter. These parameters are user-definable configuration settings that control various behaviors within the tool. Refer to `report_param` for a description of what each parameter configures or controls.

Arguments

`<name>` - (Required) The name of the parameter to get the value of. The list of user-definable parameters can be obtained with `list_param`. This command requires the full name of the desired parameter. It does not perform any pattern matching, and accepts only one parameter at a time.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example returns the current value of the MaxThreads parameter used for multi-threaded processes:

```
get_param general.MaxThreads
```

See Also

- list_param
- report_param
- reset_param
- set_param

get_partition_defs

Get a list of PartitionDefs

Syntax

```
get_partition_defs [-regexp] [-nocase] [-filter <arg>] [-quiet] [-verbose]
     [<patterns>]
```

Returns

List of PartitionDef objects.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching (valid only when -regexp specified)
[-filter]	Filter list with expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match partition definition names against patterns Default: *

Categories

Object, Partition

Description

IMPORTANT! *You must first define the project as a Partial Reconfiguration (PR) project by setting the PR_FLOW property on the project to TRUE, or by using the* **Tools → Enable Partial Reconfiguration** *command.*

Get a list of all Partition Definition (partitionDef) objects in the current design, or the partitionDefs that match a specified search pattern.

The Partial Reconfiguration flow lets you create Partition Definitions (partitionDefs) from hierarchical cells in a design, and to specify reconfigurable modules (RMs) to be assigned to these partitionDefs to create a unique configurations of the design based on the combination of the core design and one or more RMs. The PR design flow requires the implementation of each PR configuration, resulting in partial bitstreams for the RMs, but complete bitstreams for each integrated configuration. Refer to the *Vivado Design Suite User Guide: Dynamic Function eXchange (UG909)* for more information.

This command returns a list of partitionDef objects, or returns an error if the command fails.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_partition_defs` based on property values on the partitionDefs. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the "partitionDef" object, "DEFAULT_RM" and "LIBRARY" are two of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Send Feedback

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match partitionDefs against the specified search pattern. The default pattern is the wildcard '*' which gets a list of all partitionDefs in the project.

Example

The following example gets all of the partitionDef objects in the project:

```
get_partition_defs
```

See Also

- create_partition_def
- delete_partition_defs
- set_property

get_parts

Get a list of parts available in the software

Syntax

```
get_parts [-regexp] [-filter <arg>] [-of_objects <args>] [-quiet]
     [-verbose] [<patterns>]
```

Returns

List of part objects.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-filter]	Filter list with expression
[-of_objects]	Get the parts of the objects specified: project, design, or run.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match part names against patterns Default: * Values: The default search pattern is the wildcard *, or .* when -regexp is specified.

Categories

Object

Description

Gets a list of parts that match a specified search pattern. The default command gets a list of all parts.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_parts` based on property values on the parts. You can find the properties on an object with the `report_property` or `list_property` commands. Any property/value pair can be used as a filter. In the case of the part object, "DEVICE", "FAMILY" and "SPEED" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <args> - (Optional) Return the parts of the specified project, design, or run objects.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

Send Feedback

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match parts against the specified patterns. The default pattern is the wildcard '*' which gets a list of all parts. More than one search pattern can be specified to find parts based on different search criteria.

Note: You must enclose multiple search patterns in braces, {}, or quotes, "", to present the list as a single element.

Examples

The following example gets a list of 7vx485t parts, speed grade -1:

```
get_parts -filter {DEVICE =~ xc7vx485t* && speed == -1}
```

The following example gets the number of 7 series and 6 series Virtex parts:

```
llength [get_parts -regexp {xc7v.* xc6V.*} -nocase]
```

Note: If there are no parts matching the pattern, the tool will return a warning.

See Also

- list_property
- report_property

Send Feedback

get_path_groups

Get a list of path groups in the current design

Syntax

```
get_path_groups [-regexp] [-nocase] [-quiet] [-verbose] [<patterns>]
```

Returns

List of path groups.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching (valid only when -regexp specified)
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match path group names against patterns Default: *

Categories

XDC, Object

Description

Gets a list of timing path groups in the current project that match a specified search pattern. The default command gets a list of all path groups in the design.

Path groups are automatically created when a new clock is created in the design, containing all paths in that clocks domain. Path groups can also be manually created with the use of the `group_path` command.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Send Feedback

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match path groups against the specified patterns. The default pattern is the wildcard '*' which gets all path groups in the project.

Examples

The following example gets a list of all the path groups in the design.

```
get_path_groups
```

The following example gets all path groups with the string "Clk" somewhere in the name:

```
get_path_groups *Clk*
```

Note: If no path groups match the pattern you will get a warning.

See Also

- group_path

get_pblocks

Get a list of Pblocks in the current design

Syntax

```
get_pblocks [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>]
      [-include_nested_pblocks] [-quiet] [-verbose] [<patterns>]
```

Returns

List of Pblock objects.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching (valid only when -regexp specified)
[-filter]	Filter list with expression
[-of_objects]	Get Pblocks of these cells
[-include_nested_pblocks]	Display the the list of nested pblocks
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match Pblock names against patterns Default: *

Categories

Object, Floorplan, XDC

Description

Gets a list of Pblocks defined in the current project that match a specific pattern. The default command gets a list of all Pblocks in the project.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Send Feedback

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_pblocks` based on property values on the Pblocks. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the Pblock object, "NAME" and "GRID_RANGES" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` `<arg>` - (Optional) Get the Pblocks to which the specified cells are assigned.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

Send Feedback

`-include_nested_pblock` - (Optional) Also get and return any Pblocks nested inside the group of Pblocks returned by the specified search. The nested Pblocks will be added to the list of Pblock objects returned by this command. Refer to the `create_pblock` command for an example of nesting Pblocks.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match Pblocks against the specified patterns. The default pattern is the wildcard '*' which returns all Pblocks in the project.

Examples

The following example gets a list of all Pblocks in the current project:

```
get_pblocks
```

This example gets a list of all Pblocks which do not have a Slice Range defined:

```
get_pblocks -filter {GRIDTYPES !~ SLICE}
```

The following example gets the Pblock assignments of the specified cell:

```
get_pblocks -of [get_cells CORE/BR_TOP/RLD67_MUX/REG_PMBIST_C1]
```

This example returns the specified Pblock, including any nested Pblocks:

```
get_pblocks -include_nested_pblocks usbTop
```

See Also

- add_cells_to_pblock
- create_pblock
- get_cells
- resize_pblock

get_pins

Get a list of pins in the current design

Syntax

```
get_pins [-hsc <arg>] [-hierarchical] [-regexp] [-nocase] [-leaf]
    [-filter <arg>] [-of_objects <args>] [-match_style <arg>]
    [-include_replicated_objects] [-quiet] [-verbose] [<patterns>]
```

Returns

List of pin objects.

Usage

Name	Description
[-hsc]	Hierarchy separator Default: /
[-hierarchical]	Search level-by-level in current instance
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching (valid only when -regexp specified)
[-leaf]	Get leaf/global pins of nets with -of_objects
[-filter]	Filter list with expression
[-of_objects]	Get pins of these cells, nets, timing paths, clocks, drc violations
[-match_style]	Style of pattern matching, valid values are ucf, sdc Default: sdc
[-include_replicated_objects]	Include replicated objects when searching for patterns. This option is valid only when patterns is specified.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match pin names against patterns Default: *

Categories

[SDC](#), [XDC](#), [Object](#)

Description

Gets a list of pin objects in the current design that match a specified search pattern. The default command gets a list of all pins in the current_instance of the open design, as specified by the `current_instance` command. You can use the `-hierarchical` option to extract pins from the hierarchy of the current design.

Send Feedback

> **IMPORTANT!** *Because there are so many pins in the design, the `get_pins` command can cause performance issues, and add significant time to processing design constraints. In many cases, a design constraint that is written with the `get_pins` command can be rewritten using the `get_cells` command, as shown in the examples, to significantly improve constraint processing and performance of the Vivado tool.*

The `get_pins` command also includes an option to get all replicated pins that are added to a design during physical optimization, or `phys_opt_design`. The use of the `-include_replicated_objects` option returns the pins on replicated cells when the pins of an original cell are returned. You can use this option to ensure that constraints or properties that are applied to the pins of a cell are also applied to the pins of its replicated cells.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Arguments

`-hsc <arg>` - (Optional) The default hierarchy separator is '/'. Use this argument to specify a different hierarchy separator.

`-hierarchical` - (Optional) Get pins from all levels of the design hierarchy starting from the level of the current_instance, or from the top of the current design. Without this argument, the command will only get pins from the current_instance of the design hierarchy. When using `-hierarchical`, the search pattern should not contain a hierarchy separator because the search pattern is applied at each level of the hierarchy, not to the full hierarchical cell name. For instance, searching for U1/* searches each level of the hierarchy for pins with U1/ in the name. This may not return the intended results. See `get_cells` for examples of `-hierarchical` searches.

Note: When used with `-regexpr`, the specified search string is matched against the full hierarchical name, and the U1/* search pattern will work as intended.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

Send Feedback

`-leaf` - (Optional) Include leaf pins, from primitive or black box cells, for the objects specified with the `-of_object` argument.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_pins` based on property values on the pins. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the pins object, "PARENT" and "TYPE" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

`-of_objects` <arg> - (Optional) Get the pins connected to the specified cell, clock, timing path, or net; or pins associated with specified DRC violation objects.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-match_style [sdc | ucf]` - (Optional) Indicates that the search pattern matches UCF constraints or SDC constraints. The default is SDC.

`-include_replicated_objects` - (Optional) Include pins that have been added through replication of cell instances during optimization. This option is valid only when specified with <patterns>, and returns the pins matching the specified pattern, from replicated cell instances. As a default, the `get_pins` command does not return the pins of replicated cells.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match pins against the specified patterns. The default pattern is the wildcard '*' which gets a list of all pins in the project. More than one pattern can be specified to find multiple pins based on different search criteria.

Note: You must enclose multiple search patterns in braces, {}, or quotes, "", to present the list as a single element.

Examples

The following example gets a list of pins attached to the specified cells:

```
get_pins -of_objects [get_cells usb*]
```

Note: If there are no pins matching the pattern, the tool will return a warning.

This example shows how using `get_cells` can improve the performance of the `get_pins` command:

```
[get_pins -hier * -filter {NAME=~xx*/yy*}]
```

can be rewritten as:

```
[get_pins -filter {REF_PIN_NAME=~yy*} -of [get_cells -hier xx*]]
```

The following shows how rewriting the `set_max_delay` constraint can significantly improve performance:

```
set_max_delay 15 -from [get_pins -hier \
   -filter name=~*/aclk_dpram_reg*/*/CLK] \
   -to [get_cells -hier -filter name=~*/bclk_dout_reg*] -datapath_only
```

can be rewritten as:

```
set_max_delay 15 -from [get_pins -of \
   [get_cells -hier -filter name=~*aclk_dpram_reg*/*] \
   -filter {REF_PIN_NAME == CLK}] \
   -to [get_pins -of [get_cells -hier -filter {name =~ */bclk_dout_reg*}] \
   -filter {REF_PIN_NAME == D}] -datapath_only
```

TIP: *Although the second command syntax is more complex, the performance gains can be significant.*

This example runs the `report_drc` command on the current design, and then returns any pins associated with DRC violations:

```
report_drc -name drc_1
get_pins -of_objects [get_drc_violations]
```

See Also

- current_instance
- get_cells
- get_drc_violations
- list_property
- phys_opt_design
- report_drc
- report_property
- set_max_delay

get_pips

Get a list of programmable interconnect points (pips) on the current device.

Syntax

```
get_pips [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>] [-uphill]
    [-downhill] [-from <args>] [-to <args>] [-quiet] [-verbose]
    [<patterns>]
```

Returns

Pips

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-of_objects]	Get the pips of these sites, tiles, wires, nodes, pips, or nets.
[-uphill]	Get the pips uphill from the provided wire or pip.
[-downhill]	Get the pips downhill from the provided wire or pip.
[-from]	-from <pip/site pin> Return the ordered list of pips beginning at this pip or site pin. May be used in combination with uphill. Default is downhill. -all is implied.
[-to]	-to <pip/site pin> Return the ordered list of pips ending at this wire or site pin. May be used in combination with uphill. Default is downhill. -all is implied.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match pips against patterns Default: *

Categories

[Object](#), [XDC](#)

Description

Programmable interconnect points, or PIPs, provide the physical routing paths on the device used to connect logic networks. This command returns a list of PIPs on the device that match a specified search pattern. The command requires a design to be open.

The default command gets a list of all PIPs on the device. However, this is not a recommended use of the command due to the number of pips on a device. You should specify the `-of_objects` argument to limit the number of pips returned.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_pips` based on property values on the PIPs. You can find the properties on an object with the `report_property` or `list_property` commands. Any property/value pair can be used as a filter. In the case of the PIP object, "IS_DIRECTIONAL" and "FROM_PIN" are two of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <args> - (Optional) Return the PIPs of the specified sites, tiles, wires, nodes, pips, or nets objects. Xilinx recommends that you always use the `-of_objects` argument to limit the runtime and memory used by the `get_pips` command. The number of programmable interconnect points returned can be considerable.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-uphill` - (Optional) Return the PIPs uphill of the specified wire or PIPs. Uphill PIPs precede the specified object in the logic network.

`-downhill` - (Optional) Return the PIPs downhill of the specified wire or PIPs. Downhill PIPs follow the specified object in the logic network.

`-from` <args> - (Optional) Defines the starting points of the pips to get from wire or site_pin objects. This option can be combined with `-uphill`. The default is to return pips downhill of a start point. By default, all pips will be returned.

`-to` <args> - (Optional) Defines the ending points of the pips to get from wire or site_pin objects. This option can be combined with `-uphill`. The default is to return pips downhill of a start point to the specified end point. By default, all pips will be returned.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Return PIPs matching the specified search patterns. The default pattern is the wildcard '*' which gets a list of all PIPs on the device. More than one search pattern can be specified to find PIPs based on different search criteria.

Note: You must enclose multiple search patterns in braces, {}, or quotes, "", to present the list as a single element.

Examples

The following example returns the PIPs associated with the specified tile:

```
get_pips -of_object [get_tiles DSP_R_X9Y75]
```

See Also

- get_sites
- get_tiles
- get_wires
- list_property
- report_property

Send Feedback

get_pkgpin_bytegroups

Get a list of package pin byte groups.

Syntax

```
get_pkgpin_bytegroups [-regexp] [-nocase] [-filter <arg>]
     [-of_objects <args>] [-quiet] [-verbose] [<patterns>]
```

Returns

Pin_group

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-of_objects]	Get the pin_group of these package_pins, iobank, site, or port.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match pin_group against patterns Default: *

Categories

[XDC](), [Object]()

Description

Gets a list of the byte groups on the I/O banks of the current Xilinx UltraScale device.

For 7 series devices, the hierarchy of IO Banks is divided into two object types: I/O Banks and Package Pins. For Xilinx UltraScale FPGA devices, the IO Bank hierarchy includes two additional divisions: Byte groups and Nibbles.

The relationships of these objects on an UltraScale device are defined as follows:

- An iobank has 2 or 4 bytegroups.

- Each pkgpin_bytegroup has 2 nibbles, an upper and lower, and has 13 package pins.

- Each pkgpin_nibble has 6 or 7 pins, and is the upper or lower nibble of the pkgpin_bytegroup.

- A package_pin is one pin of an iobank, a pkgpin_bytegroup, or a pkgpin_nibble.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_pkgpin_bytegroup` based on property values on the objects. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the pkgpin_bytegroup object, "NAME", and "IOBANK" are two of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

Send Feedback

`-of_objects <arg>` - (Optional) Get the pkgpin_bytegroups of the specified iobank, site, port, or package_pin objects.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match pkgpin_bytegroups against the specified patterns. The default pattern is the wildcard '*' which returns all pkgpin_bytegroups. More than one pattern can be specified to find multiple pins based on different search criteria.

Examples

The following example gets a list of all pins on the package of the target device:

```
get_pkgpin_bytegroups -of [get_iobanks 44]
```

See Also

- get_iobanks
- get_package_pins
- get_pkgpin_nibbles
- list_property
- report_property

Send Feedback

get_pkgpin_nibbles

Get a list of pkgpin nibbles.

Syntax

```
get_pkgpin_nibbles [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>]
     [-quiet] [-verbose] [<patterns>]
```

Returns

Pin_nibble

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-of_objects]	Get the pin_nibble of these package_pins, iobank, site, or port.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match pin_nibble against patterns Default: *

Categories

[XDC](), [Object]()

Description

Return a list of nibbles, or half-bytes, on the I/O banks of the current Xilinx UltraScale device.

For 7 series devices, the hierarchy of IO Banks is divided into two object types: I/O Banks and Package Pins. For Xilinx UltraScale FPGA devices, the IO Bank hierarchy includes two additional divisions: Byte groups and Nibbles.

The relationships of these objects on an UltraScale device are defined as follows:

- An iobank has 2 or 4 bytegroups.

- Each pkgpin_bytegroup has 2 nibbles, an upper and lower, and has 13 package pins.

- Each pkgpin_nibble has 6 or 7 pins, and is the upper or lower nibble of the pkgpin_bytegroup.

- A package_pin is one pin of an iobank, a pkgpin_bytegroup, or a pkgpin_nibble.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_pkgpin_nibbles` based on property values on the objects. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the pkgpin_nibble object, "NAME", "IOBANK" and "TYPE" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

Send Feedback

`-of_objects <arg>` - (Optional) Get the pkgpin_nibble objects of the specified package_pins, iobank, site, or port.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match pkgpin_nibbles against the specified patterns. The default pattern is the wildcard '*' which returns all pkgpin_nibble objects on the current device.

Examples

The following example returns the Upper nibbles associated with the specified IO bank:

```
get_pkgpin_nibbles -of [get_iobanks 44] -filter {TYPE == U}
```

See Also

- get_iobanks
- get_package_pins
- get_pkgpin_bytegroups
- list_property
- report_property

Send Feedback

get_ports

Get a list of ports in the current design

Syntax

```
get_ports [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>]
    [-match_style <arg>] [-scoped_to_current_instance] [-no_traverse]
    [-prop_thru_buffers] [-quiet] [-verbose] [<patterns>]
```

Returns

List of port objects.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching (valid only when -regexp specified)
[-filter]	Filter list with expression
[-of_objects]	Get ports of these nets, instances, sites, clocks, timing paths, io standards, io banks, package pins, drc violations
[-match_style]	Style of pattern matching, valid values are ucf, sdc Default: sdc
[-scoped_to_current_instance]	Match patterns on instance pins specified using current instance, and then find top level connected ports.
[-no_traverse]	Do not traverse to find top level terminals.
[-prop_thru_buffers]	Allow propagate through buffers when traversing to find top level terminals connected to pins of scoped instance.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match port names against patterns Default: *

Categories

[SDC](), [XDC](), [Object]()

Description

Gets a list of port objects in the current design that match a specified search pattern. The default command gets a list of all ports in the design.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Send Feedback

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_ports` based on property values on the ports. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the "ports" object, "PARENT" and "TYPE" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` `<arg>` - (Optional) Get the ports connected to the specified nets, bels, sites, clocks, timing paths, io_standards, iobanks, or package_pins; or ports associated with specified DRC violation objects.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-match_style [sdc | ucf]` - (Optional) Indicates that the search pattern matches UCF constraints or SDC constraints. The default is SDC.

`-scoped_to_current_instance` - (Optional) This returns the top-level pins of the current instance. Applies the specified search <patterns> to find the instance pins on the current instance.

`-prop_thru_buffers` - (Optional) This option returns the top-level ports connected to pins on the current instance. This option can be used with the `-scoped_to_current_instance` option to propagate the search from the pins of the current instance, through buffers, to the top-level ports of the design.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match ports against the specified patterns. The default pattern is the wildcard '*' which gets a list of all ports in the project. More than one pattern can be specified to find multiple ports based on different search criteria.

Note: You must enclose multiple search patterns in braces, {}, or quotes, "", to present the list as a single element.

Examples

The following example gets a list of pins attached to the specified cell:

```
get_ports -of_objects [lindex [get_cells] 1]
```

Note: If there are no ports matching the pattern, the tool will return a warning.

The following example runs the `report_drc` command on the current design, returns the list of violations in the specified DRC report, and then returns the ports associated with any violations of the unspecified I/O Standard rule (NSTD):

```
report_drc -name drc_1
get_drc_violations -name drc_1
get_ports -of_objects [get_drc_violations -name drc_1 NSTD*]
```

Send Feedback

This example specifies a cell for the current instance, returns the pins scoped to the current instance, and returns the top-level ports connected to those pins:

```
current_instance [get_cells dac_spi*]
get_ports -scoped_to_current_instance
get_ports -scoped_to_current_instance -prop_thru_buffers
```

See Also

- current_instance
- get_cells
- get_clocks
- get_drc_violations
- get_nets
- get_timing_paths
- list_property
- report_drc
- report_property

Send Feedback

get_power_rails

Get a list of Power Rails in the current design

Syntax

```
get_power_rails [-regexp] [-nocase] [-filter <arg>] [-quiet] [-verbose]
    [<patterns>]
```

Returns

List of power rail objects.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching (valid only when -regexp specified)
[-filter]	Filter list with expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match Power rail names against patterns Default: *

Categories

XDC, Floorplan

get_pplocs

Internal TCL task for reporting PPLOCs on pins or nets: (1) -nets option returns PPLOCs on given nets; (2) -pins option returns PPLOCs on given pins; (3) if option -nets and option -pins are both not specified, it returns all PPLOCs in the design.

Syntax

```
get_pplocs -nets <args> -pins <args> [-count] [-unlocked] [-locked]
      [-level <arg>] [-quiet] [-verbose]
```

Returns

PPLOC nodes or number of PPLOCs.

Usage

Name	Description
-nets	List of nets to report its PPLOCs
-pins	List of pins to report its PPLOCs
[-count]	Count number of PPLOCs;; Do not report PPLOC or node names.
[-unlocked]	Report unlocked/unfixed PPLOCs only
[-locked]	Report locked/fixed PPLOCs only; use -level to specify locked level.
[-level]	Specify locked level; Valid values are placement and routing. Default: placement
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Report

get_pr_configurations

Get a list of partition configurations

Syntax

```
get_pr_configurations [-regexp] [-nocase] [-filter <arg>] [-quiet]
    [-verbose] [<patterns>]
```

Returns

List of Configuration objects.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching (valid only when -regexp specified)
[-filter]	Filter list with expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match partition configuration names against patterns Default: *

Categories

Object, Partition

Description

Get a list of PR configuration objects in the current project.

In the Partial Reconfiguration (PR) design flow, the PR configuration lets you specify a reconfigurable module (RM) to assign to a specific instance of a Partition Definition (partitionDef). This flow lets you create unique configurations of the design based on the combination of the core design and one or more RMs. The PR design flow requires the implementation of each PR configuration, resulting in partial bitstreams for the RMs, but complete bitstreams for each integrated configuration. Refer to the *Vivado Design Suite User Guide: Dynamic Function eXchange* (UG909) for more information.

This command returns a list of PR configuration objects, or returns an error if the command fails.

Send Feedback

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_pr_configurations` based on property values on the configurations. You can find the properties on an object with the `report_property` or `list_property` commands.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match PR configurations against the specified search pattern. The default pattern is the wildcard '*' which gets a list of all PR configurations in the project.

Example

The following example gets all of the PR configuration objects in the project:

```
get_pr_configurations
```

See Also

- create_partition_def
- create_pr_configuration
- create_reconfig_module
- delete_pr_configurations
- setup_pr_configurations

get_primitives

Get a list of available unisim primitives for a part

Syntax

```
get_primitives [-regexp] [-nocase] [-filter <arg>] [-part <arg>]
     [-retarget] [-macro] [-hierarchy] [-quiet] [-verbose] [<patterns>]
```

Returns

Primitive types

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-part]	Part to get primitives for
[-retarget]	Include primitive types that will be automatically retargeted to the current (or specified) part
[-macro]	Include primitive types that always convert into more basic, natively supported primitives, such as logic gates
[-hierarchy]	Include primitive types that will be automatically expanded into a hierarchy of leaf cells during implementation
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match primitive types against patterns Default: *

Categories

[Object](#)

Description

Get a list of all supported primitives for the specified device. This command can be used on an open elaborated, synthesized, or implemented design, in which case it will get the PART from the current design. You can also specify the `-part` option to return the primitives for any device.

By default the command always returns native primitives that can be placed on the target part without modification. The `-retarget, -macro`, and `-hierarchy` options add additional primitives to the list returned.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_primitives` based on property values on the object. You can find the properties on an object with the `report_property` or `list_property` commands.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-part` <arg> - (Optional) Get a list of primitives for the specified part.

`-retarget` - (Optional) Include primitives that are automatically re-targeted to the current or specified part.

`-macro` - (Optional) Include macro primitives that convert into more basic primitives, such as logic gates.

`-hierarchy` - (Optional) Include primitives that will be automatically expanded into a hierarchy of leaf cells during implementation.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Required) Match primitives against the specified patterns.

Examples

The following example gets the native primitives for the current part, and includes macro primitives:

```
get_primitives -macro
```

See Also

- get_cells
- get_libs
- get_lib_pins
- get_lib_cells
- get_parts
- list_property
- report_property

get_projects

Get a list of projects

Syntax

```
get_projects [-regexp] [-nocase] [-filter <arg>] [-quiet] [-verbose]
     [<patterns>]
```

Returns

List of project objects.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching (valid only when -regexp specified)
[-filter]	Filter list with expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match project names against patterns Default: *

Categories

Object, Project

Description

Gets a list of open projects that match the specified search pattern. The default gets a list of all open projects.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Send Feedback

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_projects` based on property values on the projects. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the "projects" object, "NAME", "DIRECTORY" and "TARGET_LANGUAGE" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match projects against the specified patterns. The default pattern is the wildcard '*' which gets a list of all parts. More than one pattern can be specified to find multiple projects based on different search criteria.

Examples

The following example gets a list of all open projects.

```
get_projects
```

The following example sets a variable called project_found to the length of the list of projects returned by get_projects, then prints either that projects were found or were not found as appropriate:

```
set project_found [llength [get_projects ISC*] ]
if {$project_found > 0} {puts "Project Found."} \
    else {puts "No Projects Found."}
```

Note: If there are no projects matching the pattern you will get a warning.

See Also

- create_project
- current_project
- open_project

get_property

Get properties of object

Syntax

```
get_property [-min] [-max] [-quiet] [-verbose] <name> <object>
```

Returns

Property value

Usage

Name	Description
`[-min]`	Return only the minimum value
`[-max]`	Return only the maximum value
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<name>`	Name of property whose value is to be retrieved
`<object>`	Object to query for properties

Categories

Object, PropertyAndParameter, XDC

Description

Gets the current value of the named property from the specified object or objects. If multiple objects are specified, a list of values is returned.

If the property is not currently assigned to the object, or is assigned without a value, then the `get_property` command returns nothing, or the null string. If multiple objects are queried, the null string is added to the list of values returned.

If multiple objects are passed to the `get_property` command, you can use the -min or -max options to return the smallest or greatest value of the property specified by name. This feature can be useful when setting timing constraints.

Note: For numeric properties, the min/max determination is based on numeric values. For all other properties, the determination is based on string sorting.

This command returns a value, or list of values, or returns an error if it fails.

Send Feedback

Arguments

-min - (Optional) When multiple <objects> are specified, this option examines the values of the property specified by <name>, and returns the smallest value from the list of objects. Numeric properties are sorted by value. All other properties are sorted as strings.

-max - (Optional) When multiple <objects> are specified, this option examines the values of the property specified by <name>, and returns the largest value from the list of objects. Numeric properties are sorted by value. All other properties are sorted as strings.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the set_msg_config command.

<name> - (Required) The name of the property to be returned. The name is not case sensitive.

<objects> - (Required) One or more objects to examine for the specified property.

Examples

The following example gets the NAME property from the specified cell:

```
get_property NAME [lindex [get_cells] 3]
```

The following example returns the smallest PERIOD property from the specified clock objects:

```
get_property -min PERIOD [get_clocks]
```

This example demonstrates the string based sorting of the SITE property for the specified ports:

```
get_property SITE [get_ports]
IOB_X1Y75 IOB_X1Y76 IOB_X1Y98 IOB_X1Y125 IOB_X0Y94 IOB_X1Y95 IOB_X1Y96
IOB_X1Y93 IOB_X1Y94
get_property -min SITE [get_ports]
IOB_X0Y94
get_property -max SITE [get_ports]
IOB_X1Y98
```

Note: While IOB_X1Y125 is the largest site value on the port objects, the property value IOB_X1Y98 is returned because of the sorting of the property values as strings.

See Also

- create_property
- get_cells
- get_ports
- list_property
- list_property_value
- report_property
- reset_property
- set_property

get_qor_suggestions

Get a list of available QoR suggestions

Syntax

```
get_qor_suggestions [-filter <arg>] [-quiet] [-verbose] [<IDs>]
```

Returns

List of qor suggestion objects.

Usage

Name	Description
[-filter]	Filter list with expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<IDs>]	Match suggestion names against given names

Categories

Object, Feasibility, Timing

Description

Get existing QoR suggestion objects created by the `report_qor_suggestions` command, or read into the design using `read_qor_suggestions`. The selected QoR objects can be written to a file using the `write_qor_suggestions` command. The suggestions created by the report command remain in memory until `delete_qor_suggestions` is run, or until it is overwritten by another report.

This command returns a list of existing QoR suggestion objects from the open design, or returns an error if it fails.

Arguments

`-filter <arg>` - (Optional) Filter the returned suggestion objects with the specified expression. The `-filter` argument filters the list of objects returned by the command based on property values on the objects. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the qor_suggestion object, ENABLED, CATEGORY, and AUTO are a few of properties you might use to filter the list of suggestions.

Send Feedback

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<IDs> - (Optional) Specify a list of QoR suggestion IDs to select the specified objects.

Examples

To return an individual suggestion, you can supply the full ID:

`get_qor_suggestions` RQS_CLOCK-5_1-1

Due to the naming style of suggestions, wildcard searches do NOT work directly. However you can use the `filter` command with `-regexp` to use wildcards as follows:

`filter` -regexp [get_qor_suggestions] {NAME =~ RQS_CLOCK.*}

You can also filter suggestions objects within a given category. Available categories include: Timing, Congestion, Utilization, XDC, Clocking, and Strategy. For example, the following returns suggestions related to the Timing category:

`get_qor_suggestions` -filter {CATEGORY==Timing}

See Also

- delete_qor_suggestions
- read_qor_suggestions
- report_qor_suggestions
- write_qor_suggestions

get_reconfig_modules

Get a list of ReconfigModules

Syntax

```
get_reconfig_modules [-regexp] [-nocase] [-filter <arg>]
     [-of_objects <args>] [-quiet] [-verbose] [<patterns>]
```

Returns

List of ReconfigModule objects.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching (valid only when -regexp specified)
[-filter]	Filter list with expression
[-of_objects]	Get 'reconfig_module' objects of these types: 'partition_def'.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match reconfigurable module names against patterns Default: *

Categories

[Object](), [Partition]()

Description

Get a list of reconfigurable modules (RMs) in the current design that match a specified search pattern. The default command returns a list of all RMs in the current project.

This command returns a list of RM objects, or returns an error if the command fails.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_reconfig_modules` based on property values on the RMs. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the "RM" object, "IS_GATE_LEVEL" and "PARTITION_DEF" are two of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <arg> - (Optional) Get the reconfig_modules of specified partition_def objects.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match reconfigurable modules (RMs) against the specified search pattern. The default pattern is the wildcard '*' which gets a list of all RMs in the project.

Example

The following example gets all gate level, or netlist-based RMs in the project:

```
get_reconfig_modules -filter {IS_GATE_LEVEL}
```

See Also

- create_partition_def
- create_pr_configuration
- delete_reconfig_modules
- get_partition_defs
- set_property

get_report_configs

Get a list of Configurable Report objects

Syntax

```
get_report_configs [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>]
     [-quiet] [-verbose] [<patterns>]
```

Returns

List of Configurable Report objects.

Usage

Name	Description
`[-regexp]`	Patterns are full regular expressions
`[-nocase]`	Perform case-insensitive matching (valid only when -regexp specified)
`[-filter]`	Filter list with expression
`[-of_objects]`	Get 'report' objects of these types: 'run'.
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`[<patterns>]`	Match report names against patterns Default: *

Categories

[Object](#), [Report](#)

Description

Returns a list of report objects created by the `create_report_config` command.

This command returns the list of report objects matching the search pattern and filters, or returns an error if it fails.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_report_configs` based on property values on the objects. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the "report" object, "REPORT_TYPE", "RUN_STEP" and "STATE" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <arg> - (Optional) Get the reports associated with the specified design run objects as returned by the `get_runs` command.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match report objects against the specified patterns. The default pattern is the wildcard '*' which gets a list of all report objects in the project.

Examples

The following example gets all of the report objects in the current project:

```
get_report_configs
```

The following example gets all of the report objects in the current project that are associated with the opt_design step:

```
get_report_configs -filter {RUN_STEP == opt_design}
```

See Also

- create_report_config
- delete_report_configs
- generate_reports
- get_runs

Send Feedback

get_runs

Get a list of runs

Syntax

```
get_runs [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>] [-quiet]
    [-verbose] [<patterns>]
```

Returns

List of run objects.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching (valid only when -regexp specified)
[-filter]	Filter list with expression
[-of_objects]	Get 'run' objects of these types: 'reconfig_module'.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match run names against patterns Default: *

Categories

[Object](), [Project]()

Description

Gets a list of synthesis and implementation runs in the current project that match a specified search pattern. The default command gets a list of all runs defined in the project.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Send Feedback

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_runs` based on property values on the runs. You can find the properties on an object with the `report_property` or `list_property` commands. Any property/value pair can be used as a filter. In the case of the runs object, "CONSTRSET", "IS_IMPLEMENTATION", "IS_SYNTHESIS", and "FLOW" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <arg>` - (Optional) Get the design runs for the specified `reconfig_module` objects.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match run names against the specified patterns. The default pattern is the wildcard '*' which gets a list of all defined runs in the project. More than one pattern can be specified to find multiple runs based on different search criteria.

Examples

The following example gets a list of all incomplete runs in the current project:

```
get_runs -filter {PROGRESS < 100}
```

The following example gets a list of runs matching the specified pattern:

```
get_runs imp*
```

Note: If there are no runs matching the pattern you will get a warning.

See Also

- create_run
- current_run
- get_reconfig_modules
- report_property

get_scopes

Get a list of children HDL scopes of a scope

Syntax

```
get_scopes [-filter <arg>] [-regexp] [-nocase] [-r] [-quiet] [-verbose]
    [<patterns>...]
```

Returns

Returns HDL scope objects from the given arguments.

Usage

Name	Description
[-filter]	filters <patterns> according to the specified property-matching expressions
[-regexp]	interprets <patterns> using regular expressions
[-nocase]	only when regexp is used, performs a case insensitive match
[-r]	only when a glob or regular expression pattern is used, descends recursively into children scopes to search for <patterns>
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	the pattern strings to search for scopes. Default: * (all children scopes)

Categories

[Simulation](#)

Description

Get a list of children HDL scopes of the current or specified scope

This command returns a list of scope objects, or returns an error.

Arguments

-r - (Optional) Recursively return the children scopes of the current scope.

Send Feedback

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the returned results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_scopes` based on property values on the scope objects. You can find the properties on an object with the `report_property` or `list_property` commands.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Send Feedback

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match Scope objects against the specified patterns. The default pattern is the wildcard '*' which gets a list of all Scopes that are children of the current scope. More than one pattern can be specified to find multiple Scope objects based on different search criteria.

Note: You must enclose multiple search patterns in braces {} to present the list as a single element.

Examples

The following example recursively returns all of the children scopes of the specified scope:

```
get_scopes -r /testbench/dut
```

See Also

- current_scope
- report_scopes

get_selected_objects

Get selected objects

Syntax

```
get_selected_objects [-primary] [-quiet] [-verbose]
```

Returns

List of selected objects.

Usage

Name	Description
[-primary]	Do not include objects that were selected due to selection rules
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Object, GUIControl

Description

Gets the objects currently selected in the Vivado IDE by the `select_objects` command. Can get the primary selected object and any secondary selected objects as well.

Note: This Tcl command works only when Vivado is run in GUI mode.

Primary objects are directly selected, while secondary objects are selected based on the selection rules currently defined by the **Tools → Settings** command. Refer to the *Vivado Design Suite User Guide: Using the IDE* (UG893) for more information on setting selection rules.

This command returns a Tcl list of selected objects, even a list with just one object. This can be an issue for some of the Vivado tool commands that do not accept a list of objects, such as the `current_instance` command. In this case you can use `lindex` to pass a specific object from the `get_selected_objects` list to the current_instance command:

```
current_instance [lindex [get_selected_objects] 0]
```

Send Feedback

Arguments

`-primary` - (Optional) Indicates that only the primary selected object or objects should be returned; not secondary objects. As a default `get_selected_objects` will return all currently selected objects.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example reports the properties of all currently selected objects, both primary and secondary:

```
report_property [get_selected_objects]
```

See Also

- get_highlighted_objects
- get_marked_objects
- highlight_objects
- mark_objects
- select_objects

get_simulators

Get registered simulators

Syntax

```
get_simulators [-regexp] [-nocase] [-filter <arg>] [-quiet] [-verbose]
     [<patterns>]
```

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching (valid only when -regexp specified)
[-filter]	Filter list with expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match simulator names against patterns Default: *

Categories

ToolLaunch, Simulation

Description

Get the list of simulators registered for use with the Vivado Design Suite unified simulation environment.

The Vivado Design Suite comes with some simulators pre-registered for use with the unified simulation environment. You can also register your own third-party simulators using the `register_simulator` command.

This command returns the names of registered simulators, or returns an error if it fails.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_simulators` based on the properties on the registered simulators. You can find the properties on a registered simulator object with the `report_property` or `list_property` commands. In the case of the "simulator" object, "NAME", "DESCRIPTION", and "TCLPROC.BOOTSTRAP" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match available simulators against the specified patterns. The default pattern is the wildcard '*' which gets all registered simulators.

Example

The following example returns all registered simulators:

```
get_simulators
```

See Also

- [launch_simulation](#)

get_site_pins

Get a list of site_pins.

Syntax

```
get_site_pins [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
     [-quiet] [-verbose] [<patterns>]
```

Returns

Site_pins

Usage

Name	Description
`[-of_objects]`	Get 'site_pin' objects of these types: 'site xdef_site node pin net bel_pin'.
`[-regexp]`	Patterns are full regular expressions
`[-nocase]`	Perform case-insensitive matching. (valid only when -regexp specified)
`[-filter]`	Filter list with expression
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`[<patterns>]`	Match the 'site_pin' objects against patterns. Default: *

Categories

[Object](#), [XDC](#)

Description

Returns a list of site pins of the specified site, node, logical cell pin, or net objects in an open design.

This command recommends the use of the `-of_objects` argument to prevent high run times and compute resources.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Arguments

`-of_objects` <args> - (Optional) Return the site pins of specified site, node, pin, or net objects.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_site_pins` based on property values on the site pins. You can find the properties on an object with the `report_property` or `list_property` commands. Any property/value pair can be used as a filter. In the case of the site pin object, "IS_CLOCK", "IS_DATA" and "IS_PART_OF_BUS" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match site pins against the specified patterns. The default pattern is the wildcard '*' which gets a list of all site pins of the specified objects. More than one search pattern can be specified to find site pins based on different search criteria.

Note: You must enclose multiple search patterns in braces, {}, or quotes, "", to present the list as a single element.

Examples

The following example returns the site_pins of the specified Nets:

```
get_site_pins -of_objects [get_nets *Clk]
IOB_X1Y24/I
```

The following example returns the output site_pins associated with the site SLICE_X21Y92:

```
get_site_pins -of_objects [get_sites SLICE_X21Y92] -filter {DIRECTION==OUT}
SLICE_X21Y92/A SLICE_X21Y92/AMUX SLICE_X21Y92/AQ
SLICE_X21Y92/B SLICE_X21Y92/BMUX SLICE_X21Y92/BQ
SLICE_X21Y92/C SLICE_X21Y92/CMUX SLICE_X21Y92/COUT
SLICE_X21Y92/CQ SLICE_X21Y92/D SLICE_X21Y92/DMUX
SLICE_X21Y92/DQ
```

See Also

- get_nets
- get_nodes
- get_pins
- get_sites
- list_property
- report_property

get_site_pips

Get a list of site_pips from the given object.

Syntax

```
get_site_pips [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>]
     [-quiet] [-verbose] [<patterns>]
```

Returns

Site_pips

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-of_objects]	Get the site_pips of these sites.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match site_pips against patterns Default: *

Categories

[Object](), [XDC]()

Description

Programmable interconnect points, or PIPs, provide the physical routing paths on the device used to connect logic networks. This command returns a list of PIPs on specified sites that match a specified search pattern. The command requires a design to be open.

This command requires the use of the `-of_objects` option to specify the sites to return PIPs from.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Send Feedback

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_site_pips` based on property values on the site PIPs. You can find the properties on an object with the `report_property` or `list_property` commands. Any property/value pair can be used as a filter. In the case of the PIP object, "IS_DIRECTIONAL" and "FROM_PIN" are two of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <args> - (Optional) This option can be used with the `get_sites` command to return the PIPs of specified Sites.

Send Feedback

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match PIPs against the specified patterns. The default pattern is the wildcard '*' which gets a list of all PIPs for the Sites specified by `-of_objects`. More than one search pattern can be specified to find pins based on different search criteria.

Note: You must enclose multiple search patterns in braces, {}, or quotes, "", to present the list as a single element.

Examples

The following example returns the pins of the specified BELs associated with the specified range of sites on the device:

```
get_site_pips -of_objects [get_sites SLICE_X21Y92]
```

See Also

- get_sites
- list_property
- report_property

Send Feedback

get_sites

Get a list of Sites

Syntax

```
get_sites [-regexp] [-filter <arg>] [-nocase] [-range <args>]
    [-of_objects <args>] [-quiet] [-verbose] [<patterns>]
```

Returns

List of site objects.

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-filter]	Filter list with expression
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-range]	Match site names which fall into the range. Range is defined by exactly two site names.
[-of_objects]	Get the sites of slrs, tiles, bels, pins, package_pins, ports, pblocks, nets, site_types, io_banks, cells, clock_regions or drc_violation
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match site names against patterns. Bonded sites will also match on package pin names. Default: *

Categories

[XDC](), [Object]()

Description

Gets a list of sites on the target device that match a specified search pattern. The default command gets a list of all sites on the target device.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Send Feedback

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_sites` based on property values on the sites. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the site object, "SITE_TYPE", "IS_USED", "NUM_INPUTS", and "NUM_OUTPUTS" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-range` <arg> - (Optional) Get all the sites that fall into a specified range. The range of sites must be specified with two site values, of the same SITE_TYPE, such as {SLICE_X2Y12 SLICE_X3Y15}. The SITE_TYPE of a site can be determined by the `report_property` command.

Send Feedback

Note: Specifying a range with two different types will result in an error.

`-of_objects <arg>` - (Optional) Get sites from the specified object or objects. Valid objects include: tiles, BELs, pins, package pins, ports, Pblocks, I/O Banks, cells, and clock_regions; or sites associated with specified DRC violation objects.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match sites against the specified patterns. The default pattern is the wildcard '*' which gets a list of all sites on the target device.

Examples

The following example gets a list of all sites available on the target device:

```
get_sites
```

The following example returns the number of sites that are not currently used on the device. Both command forms in the example return the same results. The second command directly evaluates the IS_USED boolean property:

```
llength [get_sites -filter {IS_USED==0}]
-or-
llength [get_sites -filter !IS_USED]
```

Note: If no sites match the pattern you will get a warning.

The following example gets all of the sites on the device, and returns the unique SITE_TYPEs:

```
set sites [get_sites]
set type {}
foreach x $sites {
    set prop [get_property SITE_TYPE $x]
    if { [lsearch -exact $type $prop] == -1 } {
        lappend type $prop
```

```
    }
}
foreach y $type {
    puts "SITE_TYPE: $y"
}
```

The following example shows three different forms for specifying the range of sites to return:

```
get_sites -range {SLICE_X0Y0 SLICE_X1Y1}
SLICE_X0Y0 SLICE_X0Y1 SLICE_X1Y0 SLICE_X1Y1
get_sites -range SLICE_X0Y0 -range SLICE_X1Y1
SLICE_X0Y0 SLICE_X0Y1 SLICE_X1Y0 SLICE_X1Y1
get_sites -range {SLICE_X0Y0:SLICE_X1Y1}
SLICE_X0Y0 SLICE_X0Y1 SLICE_X1Y0 SLICE_X1Y1
```

See Also

- get_cells
- get_drc_violations
- get_property
- list_property
- report_property

get_slrs

Get a list of slrs.

Syntax

```
get_slrs [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>] [-quiet]
      [-verbose] [<patterns>]
```

Returns

Slr

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-of_objects]	Get the slr of these device, tiles, sites, bels, sitepins, belpins, clock region, node, pip, pin, package pin, iobank, cell.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match slr against patterns Default: *

Categories

[Object](#), [XDC](#)

Description

Get a list of the super logic regions (SLRs) on the target device. On Devices that do not contain multiple SLRs, the SLR0 is returned.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Send Feedback

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_slrs` based on property values on the SLRs. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the SLR object, "NUM_CHANNELS", "NUM_SLLS" and "NUM_TILES" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <arg> - (Optional) Get the SLRs associated with the specified device, tiles, sites, bels, site_pins, bel_pins, clock_regions, nodes, pips, pins, package_pins, iobanks, or cells.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match SLRs against the specified patterns. The default pattern is the wildcard '*' which gets a list of all SLRs in the current design. More than one pattern can be specified to find multiple SLRs based on different search criteria.

Note: You must enclose multiple search patterns in braces {} to present the list as a single element.

Examples

The following example highlights each of the SLRs on the target device in a different color:

```
foreach x [get_slrs] {
    incr i
    highlight_objects -color_index $i $x
}
```

Note: If there are no cells matching the pattern you will get a warning.

The following example returns the number of super long lines (SLLs) between super logic regions on the current device:

```
get_property NUM_SLLS [get_slrs SLR0]
```

See Also

- get_property
- list_property
- report_property

get_speed_models

Get a list of speed_models in the device.

Syntax

```
get_speed_models [-of_objects <args>] [-regexp] [-nocase] [-patterns <arg>]
     [-filter <arg>] [-speed_pattern <arg>] [-quiet] [-verbose]
```

Returns

Speed_models

Usage

Name	Description
[-of_objects]	Get 'speed_model' objects of these types: 'node bel pip cell'.
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-patterns]	Match the 'speed_model' objects against patterns. Default: *
[-filter]	Filter list with expression
[-speed_pattern]	Returns a single model that exactly matches the pattern.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Object](), [XDC]()

Description

Get speed models for UltraScale architecture device resources in the current design.

Speed files are provided by Xilinx for each device and speed grade. Speed files contain speed models. There are speed models for the various elements of a device: nodes, pips, bels. There are speed models for setup and hold, propagation delays, jitter, etc.

The speed models include information on the delays in nanoseconds (ns) associated with device resources like BELs and SITEs and routing resources. Speed models are used by the Vivado timing engine to perform analysis of the current design in the context of the target part.

The objects returned are the speed models associated with specific physical resources like pips and wires, drawn directly from the speed files. These can include capacitance and resistance values and buffer models.

Send Feedback

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

This command returns the specified speed model objects, or returns an error if the command fails.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_speed_models` based on property values on the objects. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the speed_model object, "NAME", "TYPE" and "DELAY" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <arg>` - (Optional) Get the speed models for the specified node, bel, pip, or cell objects.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example reports the properties on the speed model for an A6LUT:

```
report_property -all [lindex [get_speed_models -of \
[get_bels SLICE_X0Y0/A6LUT]] 0]
Property              Type     Read-only  Value
CLASS                 string   true       speed_model
DELAY                 double   true       0.043
FAST_MAX              double   true       0.035
FAST_MIN              double   true       0.028
IS_INSTANCE_SPECIFIC  bool     true       1
NAME                  string   true       bel_d_lut6_a1_o6
NAME_LOGICAL          string   true       bel_d_lut6_a1_o6
SLOW_MAX              double   true       0.043
SLOW_MIN              double   true       0.036
SPEED_INDEX           int      true       65535
TYPE                  string   true       bel_delay
```

The following example returns the delays, in nanoseconds, for a specific A6LUT on the device, followed by the delay name for the specified object:

```
set x [get_speed_models -of [get_bels SLICE_X0Y0/A6LUT]]
puts [format "%6.3f : %s" [get_property DELAY [lindex $x 0 ]] \
[get_property NAME [lindex $x 0 ]]]
```

See Also

- get_bels
- list_property

- report_property

get_stacks

Get list of processes in a design, which are waiting inside a subprogram

Syntax

```
get_stacks [-of_instance <arg>] [-quiet] [-verbose]
```

Returns

Returns HDL scope objects (from the given arguments) which are processes waiting in a subprogram.

Usage

Name	Description
`[-of_instance]`	Default: NULL
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution

Categories

[Simulation](#)

Description

Returns list of HDL scopes for all the processes in a design which are waiting inside a subprogram. With `-of_instance` switch, the output can be limited to such processes of the given instance only.

Arguments

`-of_instance` <HDL_Instance_scope> - (Optional) Get the list of processes (waiting inside a sub-program) for the specified HDL instances only.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Send Feedback

Examples

The following example gets a list of all sites available on the target device:

```
module top;
     int i;
     function void f(input int in1);
         automatic int a;
         a = in1 + 7;
         $display($time, " in f :: a %d in1 %d ", a, in1);
     endfunction
     task automatic t(input int in2);
         int b;
         b = in2 + 10;
         $display($time, " in t :: in2 %d b %d ", in2, b);
         #5;
         f(b);     // Case C
         $display($time, " Back in t : after wait and f(%d) ", b);
     endtask
     initial begin                         // "/top/Initial18_0"
         $display($time, " in initial 1 ");
         i = 200;
         t(i);          // Case B
         $display($time, " Back in initial 1 after t(%d) ", i);
     end
     initial begin                         // "/top/Initial25_1"
         $display($time, " in initial 2 ");
         #2;
         f(50);        // Case A
         $display($time, " Back in initial 2 after f(50) ");
     end
  endmodule
```

When simulation is stopped inside function "f" for its call at 'Case A' , the two processes `/top/Initial18_0`, and `/top/Initial25_1`, are waiting inside task "t" (call at 'CaseB') and function "f" (call at 'CaseA') respectively.

```
1. > get_stacks
      /top/Initial18_0 /top/Initial25_1
```

See Also

- report_stacks

Send Feedback

get_template_bd_designs

Get a list of IPI example designs

Syntax

```
get_template_bd_designs [-quiet] [-verbose]
```

Returns

List of IPI design objects.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

IPIntegrator

Description

The command returns the list of template block designs available in the current release of the Vivado Design Suite, or returns an error if it fails.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the set_msg_config command.

Examples

The following example returns the list of available block design templates in the current release:

```
get_template_bd_designs
```

See Also

- instantiate_template_bd_design

get_tiles

Get a list of tiles.

Syntax

```
get_tiles [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>] [-quiet]
    [-verbose] [<patterns>]
```

Returns

Tiles

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-of_objects]	Get the tiles of these slr, sites, bels, site_pins, bel_pins, nodes, wires, pips, nets, clock_regions.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match tiles against patterns Default: *

Categories

Object, [XDC](XDC)

Description

This command returns a list of tiles on the device in an open design. The default command gets a list of all tiles on the device.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_tiles` based on property values on the tile objects. You can find the properties on an object with the `report_property` or `list_property` commands. Any property/value pair can be used as a filter. In the case of the tile object, "NUM_ARCS", "NUM_SITES", and "IS_GT_SITE_TILE" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <args> - (Optional) Can be used to return the tiles associated with specified sites, bels, site_pins, bel_pins, nodes, wires, pips, nets, clock_regions, or slrs.

Send Feedback

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match tiles against the specified patterns. The default pattern is the wildcard '*' which gets a list of all tiles on the device. More than one search pattern can be specified to find tiles based on different search criteria.

Note: You must enclose multiple search patterns in braces, {}, or quotes, "", to present the list as a single element.

Examples

The following example returns the total number of tiles where the number of timing arcs is greater than 100 and 150 respectively:

```
llength [get_tiles -filter {NUM_ARCS>100} ]
13468
llength [get_tiles -filter {NUM_ARCS>150} ]
11691
```

See Also

- get_bels
- get_nodes
- get_pips
- get_site_pins
- get_sites
- get_wires
- list_property
- report_property

Send Feedback

get_timing_arcs

Get a list of timing arcs

Syntax

```
get_timing_arcs [-from <args>] [-to <args>] [-filter <arg>]
    [-of_objects <args>] [-quiet] [-verbose]
```

Returns

List of timing arc objects.

Usage

Name	Description
[-from]	List of pin or ports
[-to]	List of pin or ports
[-filter]	Filter list with expression
[-of_objects]	Get timing arcs for these cells
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[XDC](), [Object](), [Timing]()

Description

Gets a list of timing arcs for the specified objects. You can filter the timing arcs according to specified properties.

Timing arcs are a part of a timing path. A timing arc can be a wire between two pins, or can be the internal path of a logic instance between an input pin and output pin, or clock input and data output pins.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Arguments

`-from` <args> - (Optional) The starting points of the timing arcs to be returned. Ports, pins, or nets can be specified as startpoints.

`-to` <args> - (Optional) The endpoints or destination objects of timing arcs to be returned. Ports, pins, or nets can be specified as endpoints.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_timing_arcs` based on property values on the timing arcs. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the "timing arc" object, "FROM_PIN", "TO_PIN" and "LIB_CELL" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <args> - (Optional) Get timing arcs from the Specified cell objects. If a cell is specified, all cell_arcs of that cell are returned.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example returns the timing arc from the output pin of the specified buffer:

```
report_property -all [get_timing_arcs -of_objects [get_cells go_IBUF_inst]]
```

The following example returns the timing arcs of the specified cell:

```
get_timing_arcs -of_objects [get_cells count_reg[6]]
{count_reg[6]/C --> count_reg[6]/Q [Reg Clk to Q] }
{count_reg[6]/C --> count_reg[6]/D [setup] }
{count_reg[6]/C --> count_reg[6]/D [hold] }
{count_reg[6]/C --> count_reg[6]/CLR [recovery] }
{count_reg[6]/C --> count_reg[6]/CE [hold] }
{count_reg[6]/C --> count_reg[6]/CLR [removal] }
{count_reg[6]/C --> count_reg[6]/CE [setup] }
{count_reg[6]/CLR --> count_reg[6]/Q [Reg Set/Clr] }
```

See Also

- report_timing
- get_msg_config
- set_msg_config

get_timing_paths

Get timing paths

Syntax

```
get_timing_paths [-from <args>] [-rise_from <args>] [-fall_from <args>]
    [-to <args>] [-rise_to <args>] [-fall_to <args>] [-through <args>]
    [-rise_through <args>] [-fall_through <args>] [-delay_type <arg>]
    [-setup] [-hold] [-max_paths <arg>] [-nworst <arg>] [-unique_pins]
    [-slack_lesser_than <arg>] [-slack_greater_than <arg>] [-group <args>]
    [-no_report_unconstrained] [-user_ignored] [-routable_nets]
    [-sort_by <arg>] [-filter <arg>] [-regexp] [-nocase] [-cell <args>]
    [-quiet] [-verbose]
```

Usage

Name	Description
[-from]	From pins, ports, cells or clocks
[-rise_from]	Rising from pins, ports, cells or clocks
[-fall_from]	Falling from pins, ports, cells or clocks
[-to]	To pins, ports, cells or clocks
[-rise_to]	Rising to pins, ports, cells or clocks
[-fall_to]	Falling to pins, ports, cells or clocks
[-through]	Through pins, ports, cells or nets
[-rise_through]	Rising through pins, ports, cells or nets
[-fall_through]	Falling through pins, ports, cells or nets
[-delay_type]	Type of path delay: Values: max, min, min_max, max_rise, max_fall, min_rise, min_fall Default: max
[-setup]	Get max delay timing paths (equivalent to -delay_type max)
[-hold]	Get min delay timing paths (equivalent to -delay_type min)
[-max_paths]	Maximum number of paths to return: Value >=1 Default: 1
[-nworst]	List N worst paths to endpoint: Value >=1 Default: 1
[-unique_pins]	for each unique set of pins, show at most 1 path per path group
[-slack_lesser_than]	Include paths with slack less than this Default: 1e+30
[-slack_greater_than]	Include paths with slack greater than this Default: -1e+30
[-group]	Limit paths in this group(s)
[-no_report_unconstrained]	Do not get unconstrained paths
[-user_ignored]	only report paths which have infinite slack because of set_false_path or set_clock_groups timing constraints
[-routable_nets]	store the number of routable nets traversed as a property on timing paths.
[-sort_by]	Sorting order of paths: Values: group, slack Default: slack
[-filter]	Filter list with expression
[-regexp]	Patterns specified in filter are full regular expressions

Send Feedback

Name	Description
`[-nocase]`	Perform case-insensitive matching for patterns specified in filter (valid only when -regexp specified)
`[-cell]`	run get_timing_paths on the cell
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution

Categories

Object, Timing

Description

Gets timing path objects that meet the specified criteria. This command can be used to predefine timing paths to pass to the `report_timing` command for instance. Another usage of this command is to create custom reporting and analysis.

The `get_timing_paths` command is very similar to the `report_timing` command. However, `get_timing_paths` returns timing path objects which can be queried for properties, or passed to other Tcl commands for processing, where `report_timing` returns a file or a string.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Arguments

`-from` <args> - (Optional) Defines the starting points of the timing paths to be analyzed. Ports, pins, or cells can be specified as timing path startpoints. You can also specify a clock object, and all startpoints clocked by the named clock will be analyzed.

`-rise_from` <args> - (Optional) Similar to the `-from` option, but only the rising edge of signals coming from the startpoints are considered for timing analysis. If a clock object is specified, only the paths launched by the rising edge of the clock are considered as startpoints.

`-fall_from` <args> - (Optional) Similar to the `-from` option, but only the falling edge of signals coming from the startpoints are considered for timing analysis. If a clock object is specified, only the paths launched by the falling edge of the clock are considered as startpoints.

`-to` <args> - (Optional) Specifies the endpoints, or destination objects of timing paths to be analyzed. Ports, pins, and cell objects can be specified as endpoints. A clock object can also be specified, in which case endpoints clocked by the named clock are analyzed.

`-rise_to` <args> - (Optional) Similar to the `-to` option, but only the rising edge of signals going to the endpoints is considered for timing analysis. If a clock object is specified, only the paths captured by the rising edge of the named clock are considered as endpoints.

`-fall_to` <args> - (Optional) Similar to the `-to` option, but only the falling edge of signals going to the endpoints is considered for timing analysis. If a clock object is specified, only the paths captured by the falling edge of the named clock are considered as endpoints.

`-through` <args> - (Optional) Specifies that only paths through the specified pins, cell instance, or nets are considered during timing analysis. You can specify individual `-through` (or `-rise_through` and `-fall_through`) points in sequence to define a specific path through the design for analysis. The order of the specified through points is important to define a specific path. You can also specify through points with multiple objects, in which case the timing path can pass through any of the specified through objects.

`-rise_through` <args> - (Optional) Similar to the `-through` option, but timing analysis is only performed on paths with a rising transition at the specified objects.

`-fall_through` <args> - (Optional) Similar to the `-through` option, but timing analysis is only performed on paths with a falling transition at the specified objects.

`-delay_type` <arg> - (Optional) Specifies the type of delay to analyze when running the timing report. The valid values are min, max, min_max, max_rise, max_fall, min_rise, min_fall. The default setting for `-delay_type` is max.

`-setup` - (Optional) Check for setup violations. This is the same as specifying `-delay_type` max.

`-hold` - (Optional) Check for hold violations. This is the same as specifying `-delay_type min`.

Note: `-setup` and `-hold` can be specified together, which is the same as specifying `-delay_type` min_max.

`-max_paths` <arg> - (Optional) The maximum number of paths to output when sorted by slack; or the maximum number of paths per path group when sorted by group, as specified by `-sort_by`. This is specified as a value greater than or equal to 1. The default value is 1, returning the single worst timing path, or the worst path per group.

`-nworst` <arg> - (Optional) The number of timing paths to show to each endpoint. The timing report will report the N worst paths based on the specified value. This is specified as a value greater than or equal to 1. The default setting is 1.

`-unique_pins` - (Optional) Show only one timing path per each unique pin or group of pins. This is a boolean option, enabled by its use.

`-routable_nets` - (Optional) Report the number of routable nets traversed as a property of the timing paths returned.

`-slack_greater_than <arg>` - (Optional) Report timing on paths with a calculated slack value greater than the specified value. Used with `-slack_lesser_than` to provide a range of slack values of specific interest.

`-slack_lesser_than <arg>` - (Optional) Report timing on paths with a calculated slack value less than the specified value. Used with `-slack_greater_than` to provide a range of slack values of specific interest.

`-group <args>` - (Optional) Report timing for paths in the specified path groups.

`-no_report_unconstrained` - (Optional) Do not report timing on unconstrained paths.

`-user_ignored` - (Optional) Show the timing paths that are ignored during timing analysis because the user has specified `set_false_path` or `set_clock_groups` timing constraints. This is a boolean option, enabled by its use.

`-sort_by [ slack | group ]` - (Optional) Sort timing paths by path groups, or by slack values. The default sort order is by slack values.

`-filter <args>` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_timing_paths` based on property values on the timing paths. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the timing path object, "DATAPATH_DELAY", "ENDPOINT_PIN" and "ENDPOINT_CLOCK" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

The following example gets the first 100 most critical timing paths objects and returns only those from the path group clk_tx_clk_core_1:

```
get_timing_paths -max_paths 100 -filter {GROUP == clk_tx_clk_core_1}
```

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Send Feedback

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-cell <arg>` - (Optional) Apply the `get_timing_paths` command to the specified cell. Cells can be specified by name, or as an object returned by the `get_cells` command.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

Get the five worst timing paths from the specified endpoint and report all the properties of the fourth timing path in the list:

```
report_property -all [lindex [get_timing_paths -to [get_ports led_pins[*]]\
-nworst 5] 3]
```

Define a procedure called custom_report and report the 100 worst paths from the clk_tx_clk_core_1 path group using the procedure.

```
proc custom_report { listOfPaths } {
   puts [format {%-40s %-40s %-20s %-20s %7s} "Startpoint" "Endpoint" \
      "Launch Clock" "Capture Clock" "Slack"]
   puts [string repeat "-" 140]
   foreach path $listOfPaths {
      set startpoint [get_property STARTPOINT_PIN $path]
      set startclock [get_property STARTPOINT_CLOCK $path]
      set endpoint [get_property ENDPOINT_PIN $path]
```

```
        set endclock [get_property ENDPOINT_CLOCK $path]
        set slack [get_property SLACK $path]
        puts [format {%-40s %-40s %-20s %-20s %7s} $startpoint $endpoint \
            $startclock $endclock $slack]
    }
}
set paths [get_timing_paths -group clk_tx_clk_core_1 -max_paths 100]\
custom_report $path
```

Use get_timing_paths to perform a timing analysis on the specified paths.

```
set paths [get_timing_paths -group clk_tx_clk_core_1 -max_paths 100]
report_timing -of_objects $paths
```

Which is the equivalent of:

```
report_timing -group clk_tx_clk_core_1 -max_paths 100
```

Return timing paths where the logic levels are greater than the specified number of logic levels.

```
get_timing_paths -max_paths 1000 -filter {LOGIC_LEVELS > 1}
```

See Also

- report_property
- report_timing

Send Feedback

get_value

Get current value of the selected HDL object (variable, signal, wire, reg)

Syntax

```
get_value [-radix <arg>] [-quiet] [-verbose] <hdl_object>
```

Returns

Returns a string representation of value of a hdl_object.

Usage

Name	Description
[-radix]	radix specifies the radix to use for printing the values of the hdl_objects. Allowed values are: default, dec, bin, oct,hex, unsigned, ascii, smag
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hdl_object>	The hdl_object to retrieve the current value

Categories

Simulation

Description

Get the value of a single HDL object at the current simulation run time.

> 💡 **TIP:** *Use the* `report_values` *command to return the values of more than one HDL objects.*

HDL objects include HDL signals, variables, or constants as defined in the Verilog or VHDL test bench and source files. An HDL signal includes Verilog wire or reg entities, and VHDL signals. Examples of HDL variables include Verilog real, realtime, time, and event.

HDL constants include Verilog parameters and localparams, and VHDL generic and constants. The HDL scope, or scope, is defined by a declarative region in the HDL code such as a module, function, task, process, or begin-end blocks in Verilog. VHDL scopes include entity/architecture definitions, block, function, procedure, and process blocks.

Arguments

`-radix <arg>` - (Optional) Specifies the radix to use when returning the value of the specified object. Allowed values are: `default`, `dec`, `bin`, `oct`, `hex`, `unsigned`, `ascii`, or `smag`.

Send Feedback

Note: The radix `dec` indicates a signed decimal. Specify the radix `unsigned` when dealing with unsigned data.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hdl_object> - (Required) Specifies a single HDL object to get the value of. The object can be specified by name, or can be returned as an object from the `get_objects` command.

Examples

The following example gets the value of the sysClk signal:

```
get_value sysClk
    Z
```

This example shows the difference between the `bin`, `dec`, and `unsigned` radix on the value returned from the specified bus:

```
get_value -radix bin /test/bench_VStatus_pad_0_i[7:0]
    10100101
get_value -radix unsigned /test/bench_VStatus_pad_0_i[7:0]
    165
get_value -radix dec /test/bench_VStatus_pad_0_i[7:0]
    -91
```

See Also

- current_time
- get_objects
- set_value
- report_values

get_waivers

Get one or more DRC/METHODOLOGY/CDC/LINT message waivers

Syntax

```
get_waivers [-type <arg>] [-id <arg>] [-of_objects <args>] [-regexp]
    [-filter <arg>] [-nocase] [-quiet] [-verbose] [<patterns>]
```

Returns

Waivers

Usage

Name	Description
[-type]	Type of waiver - DRC, METHODOLOGY, CDC, LINT, ALL
[-id]	ID of the DRC/METHODOLOGY/CDC/LINT message being waived
[-of_objects]	List of objects (cells, nets, pins, sites, etc.) for which DRC/ METHODLOGY/CDC/LINT waiver(s) were set
[-regexp]	Patterns are full regular expressions
[-filter]	Filter list with expression
[-nocase]	Perform case-insensitive matching (valid only when -regexp specified)
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match waiver names against patterns Default: * Values: The default search pattern is the wildcard *, or .* when -regexp is specified.

Categories

[Waiver](), [Object]()

Description

The `create_waiver` command lets you select individual DRC, methodology, or CDC violations or rule checks that can be waived for a design, letting you move forward in the design flow. The `get_waivers` command lets you query the defined waiver objects in the current design.

A waiver must be specified for an individual DRC or methodology violation, or for a specific DRC or methodology check, or for a CDC path. The waiver must be assigned to a specific object, or specific violation ID, or for paths using `-from/-to` arguments. You can format the `get_waivers` command to return the specific types of waivers you are looking for, or waivers associated with specific objects.

Send Feedback

You can report the waivers defined in the current design with `report_waivers`, and remove waivers from the design using `delete_waivers`.

Arguments

`-type <arg>` - (Optional) Specifies the type of waiver to get. Currently supports DRC, METHODOLOGY, and CDC.

`-id <arg>` - (Optional) Specifies the ID of the check or violation associated with the waiver.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_waivers` based on property values on the waivers. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the waivers object, "NAME", "OBJECT_COUNTS" and "TYPE" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <arg>` - (Optional) Get the waivers associated with the specified objects (cells, nets, pins, sites, etc.).

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match waivers against the specified search pattern. The default pattern is the wildcard '*' which gets a list of all waivers in the project.

Examples

This example gets any waivers in the current design:

```
get_waivers
```

The following example gets all DRC check waivers:

```
get_waivers -type DRC *
```

The following example gets all waivers associated with the specified objects:

```
get_waivers -of_objects [get_ports {src_in* dest_out*}]
```

See Also

- create_waiver
- delete_waivers
- report_waivers

get_wave_configs

Gets the wave configs that match the given options

Syntax

```
get_wave_configs [-regexp] [-nocase] [-filter <arg>] [-quiet] [-verbose]
    [<patterns>...]
```

Returns

Wave configs that match the given options.

Usage

Name	Description
[-regexp]	interprets <patterns> using regular expressions
[-nocase]	only when regexp is used, performs a case insensitive match
[-filter]	filters <patterns> according to the specified property-matching expressions
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	the pattern strings to search for wave configuration names

Categories

Waveform

Description

Get the wave configuration objects that match the specified search options in the current simulation.

In the Vivado® simulator GUI, you can work with a waveform to analyze your design and debug your code. The Wave Config file contains the list of wave objects (signals, dividers, groups, virtual buses) to display, and their display properties, plus markers. A wave configuration displays with top-level HDL objects, and can be further populated using commands like `add_wave` and `add_wave_divider`.

This command returns the matching wave configuration objects, or returns nothing if no objects matched the search pattern.

Send Feedback

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_wave_configs` based on property values on the wave configuration objects. You can find the properties on an object with the `report_property` or `list_property` commands. In the case of the "Wave Configuration" object, "NAME", "NEEDS_SAVE", and "FILE_PATH" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Send Feedback

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Match wave configuration objects in the current simulation against the specified search patterns. The default pattern is the wildcard '*' which gets a list of all open wave configurations in the simulation.

Examples

The following example returns all wave configuration objects, in the current simulation, that have unsaved changes:

```
get_wave_config -filter {NEEDS_SAVE}
```

See Also

- close_wave_config
- create_wave_config
- current_wave_config
- open_wave_config
- save_wave_config

get_waves

Gets wave objects from a wave configuration

Syntax

```
get_waves [-of <args>] [-regexp] [-nocase] [-filter <arg>] [-recursive]
    [-r] [-long_name] [-short_name] [-quiet] [-verbose] <patterns>...
```

Returns

A collection of found wave objects.

Usage

Name	Description
[-of]	the wave configuration, group, or virtual bus to search
[-regexp]	interprets <patterns> using regular expressions
[-nocase]	only when regexp is used, performs a case insensitive match
[-filter]	filters <patterns> according to the specified property-matching expressions
[-recursive]	if the design object is a scope, this option specifies that wave objects for all design objects under that scope should be created
[-r]	if the design object is a scope, this option specifies that wave objects for all design objects under that scope should be created
[-long_name]	search wave objects using the long form of their names
[-short_name]	search wave objects using the short form of their names
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<patterns>	the design objects from which to create wave objects

Categories

[Waveform](#)

get_wires

Get a list of wires.

Syntax

```
get_wires [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>]
    [-uphill] [-downhill] [-from <args>] [-to <args>] [-quiet] [-verbose]
    [<patterns>]
```

Returns

Wires

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-of_objects]	Get the wires of these tiles, nodes, pips, or nets.
[-uphill]	Get the wires uphill from the provided pip.
[-downhill]	Get the wires downhill from the provided pip.
[-from]	-from <pip/site pin> Return the ordered list of wires beginning at this pip or site pin. May be used in combination with uphill. Default is downhill. -all is implied.
[-to]	-to <pip/site pin> Return the ordered list of wires ending at this wire or site pin. May be used in combination with uphill. Default is downhill. -all is implied.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match wires against patterns Default: *

Categories

Object, XDC

Description

Returns a list of wires in the design that match a specified search pattern in an open design.

The default command gets a list of all wires in the design.

Note: To improve memory and performance, the `get_*` commands return a container list of a single type of objects (e.g. cells, nets, pins, or ports). You can add new objects to the list (using `lappend` for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error.

Arguments

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter` <args> - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by `get_wires` based on property values on the wires. You can find the properties on an object with the `report_property` or `list_property` commands. Any property/value pair can be used as a filter. In the case of the wire object, "NAME", "NUM_DOWNHILL_PIPS" and "NUM_UPHILL_PIPS" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

Send Feedback

-of_objects <args> - (Optional) Return the wires of the specified nodes, PIPs, or tiles.

Note: The -of_objects option requires objects to be specified using the get_* commands, such as get_cells or get_pins, rather than specifying objects by name. In addition, -of_objects cannot be used with a search <pattern>.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the set_msg_config command.

<patterns> - (Optional) Return wires matching the specified search patterns. The default pattern is the wildcard '*' which gets a list of all wires in the design. More than one search pattern can be specified to find wires based on different search criteria.

Note: You must enclose multiple search patterns in braces, {}, or quotes, "", to present the list as a single element.

Examples

The following example returns the wires associated with the specified tile:

```
get_wires -of_objects [get_tiles IO_INT_INTERFACE_L_X0Y198]
```

See Also

- get_nodes
- get_pips
- get_tiles
- list_property
- report_property

Send Feedback

group_bd_cells

Create a hierarchical cell, and then move the group of cells into the hierarchy cell. The connections between these cells are maintained; the connections between these cells and other cells are maintained through crossing hierarchy cell.

Syntax

```
group_bd_cells [-prefix <arg>] [-quiet] [-verbose] [<target_cell_name>]
    [<cells>...]
```

Returns

0 if success.

Usage

Name	Description
[-prefix]	Prefix name to add to cells
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<target_cell_name>]	Target cell
[<cells>]	Match engine names against cell names Default: *

Categories

IPIntegrator

Description

Create a new hierarchical module in the current IP integrator subsystem design, and move the specified cells into that module.

You can also optionally move a group of specified cells into the hierarchical module. The connections between the specified cells are maintained. The connections between the cells being moved are maintained; connections between these cells and other cells that are not being moved are maintained automatically by IP integrator adding pins and ports to cross the hierarchical boundary.

You can also move cells into the hierarchical module by using the `move_bd_cells` command after the hierarchical module has been created using the `create_bd_cells` command.

The command returns the name of the created hierarchical module if successful, or an error message if it fails.

Send Feedback

Arguments

`-prefix <arg>` - (Optional) A prefix name to apply to any cells that are moved into the hierarchical module.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<target_cell_name>` - (Required) The name to assign to the hierarchical module that is created.

`<cells>` - (Required) The list of cells, specified by the `get_bd_cells` command, to move from the current IP subsystem design into the hierarchical module.

Example

The following example creates a hierarchical block in the current IP integrator subsystem design, and moves the three specified cells into the block assigning them a name prefix as indicated:

```
group_bd_cells -prefix M1_ module1 [get_bd_cells /microblaze_1_xlconcat] \
[get_bd_cells /microblaze_1_axi_intc] [get_bd_cells /proc_sys_reset_1]
```

See Also

- create_bd_cell
- get_bd_cells
- move_bd_cells

group_path

Groups paths for cost function calculations

Syntax

```
group_path [-name <args>] [-weight <arg>] [-default] [-from <args>]
    [-to <args>] [-through <args>] [-quiet] [-verbose]
```

Usage

Name	Description
[-name]	The name of group, can be multiple names
[-weight]	Cost function Weight, Valid values are 1, 2 Default: 1.0
[-default]	Restore path to its default group
[-from]	Filter by paths starting at these path startpoints
[-to]	Filter by paths terminating at these path endpoints
[-through]	Consider paths through pins, cells or nets
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[SDC](), [XDC]()

Description

This command lets you group a set of paths for cost function calculations, primarily for timing analysis. The Vivado tool automatically defines path groups of clock signals for special handling. User-defined path groups can be specified generally as from a startpoint, or to an endpoint, or as specific paths from-through-to specific points. Once a path group has been created, some timing analysis can be performed against it with the `report_timing` command.

You can specify a weight for an existing path group of clocks, and let the placement, routing, and optimization engines prioritize those paths first.

To remove a path from a path group, you must specify the `-default` option to remove the path from a named path group and restore the path to the standard "default" path group.

This option has the following limitations:

- For paths originally assigned by the tool to a clock path group, the `-default` option will revert those paths to the clock path group instead of reverting them to the "default" path group.

Send Feedback

- `group_path -default` and `reset_path` are completely independent commands. The `reset_path` command doesn't affect path groups, and the `group_path` command doesn't affect other timing exceptions .

The path groups currently defined in a design can be found by using the `get_path_groups` command.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

`-name <arg>` - (Optional) Specifies the name of the path group. If the path group name already exists, the specified paths will be added to the existing group.

`-weight [ 1 | 2 ]` - (Optional) Specify the priority of real clock path groups with a value of 1 for standard priority, or 2 for high priority. High priority path groups are processed first during placement, routing, and physical optimization. The default is 1, or standard priority.

> **IMPORTANT!** *The `-weight` option is only supported for use with clock path groups, and requires the use of `-name` to specify the path group. It is not supported for use with user-defined path groups, or with `-from/-through/-to` options.*

`-default` - (Optional) Restores a path group back to the "default" group, which contains all paths not assigned to other group. This option cannot be specified with `-name` or `-weight`. The path must be specified by `-from/-through/-to` as it was initially defined when assigned to a path group.

`-from` <args> - (Optional) Include paths starting at the specified startpoints. The startpoints can be specified as pins, ports, or clocks.

`-through` <element_names> - (Optional) Include paths routed through the specified pins, cells, or nets.

`-to` <path_names> - (Optional) Include all paths to the specified endpoints. Endpoints can be specified as pins, ports, or clocks.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example creates a group named signal_grp to the specified registers' endpoints matching *signal*reg/D, and then reports timing on the specified group:

```
group_path -to [get_pins *signal*reg/D -hierarchical] -name signal_grp
report_timing -group signal_grp
```

The path group signal_grp is also returned by the `get_path_groups` command:

```
get_path_groups signal_grp
```

The following example removes the path from the signal_grp, restoring it to the default path group:

```
group_path -to [get_pins *signal*reg/D -hierarchical] -default
```

See Also

- get_path_groups
- report_timing

help

Display help for one or more topics

Syntax

```
help [-category <arg>] [-args] [-syntax] [-long] [-prop <arg>]
    [-class <arg>] [-message <arg>] [-quiet] [-verbose]
    [<pattern_or_object>]
```

Usage

Name	Description
`[-category]`	Search for topics in the specified category
`[-args]`	Display arguments description
`[-syntax]`	Display syntax description
`[-long]`	Display long help description
`[-prop]`	Display property help for matching property names Default: *
`[-class]`	Display object type help
`[-message]`	Display information about the message with the given message. Every message delivered by the tool has a unique global message ID that consists of an application sub-system code and a message identifier. Example: -message {Common 17-8}.
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`[<pattern_or_object>]`	Display help for topics that match the specified pattern Default: *

Categories

Project

Description

Returns a long description of the specified Tcl command; or a list of available Xilinx Tcl command categories; or a list of commands matching a specific pattern.

The default `help` command without any arguments returns a list of Tcl command categories that can be further explored. Command categories are groups of commands performing a specific function, like File I/O commands for instance.

Available options for the `help` command can return just the command syntax for a quick reminder of how the command should be structured; the command syntax and a brief description of each argument; or the long form of the command with more detailed descriptions and examples of the command.

Send Feedback

To limit the memory usage of the Vivado Design Suite, some features of the tool are only loaded into memory when that feature set is used. To access the complete list of Tcl commands and help text associated with a given feature, you must load the feature into memory using the `load_features` command.

The help command can also return any available information related to various properties assignable to design objects. Use the `-prop` and `-class` options to return help information for properties.

This command returns the specified help text, or an error.

Arguments

`-category <arg>` - (Optional) Get a list of the commands grouped under the specified command category.

`-args` - (Optional) Get abbreviated help text for the specified command. The default is to return the extended help for the specified command. Use this argument to keep it brief.

`-syntax` - (Optional) Returns only the syntax line for the command as a quick reminder of the proper form for using the command.

`-long` - (Optional) Returns the extended help description for the command, including the syntax, a brief description of the arguments, and a more detailed description of the command with examples. This is the default setting for the help command.

`-prop <arg>` - (Optional) Return the specified property of an object class, or the properties assigned to a specific object in the current design.

Note: This option requires the use of `-class`, or the specification of a single design object.

`-class <arg>` - (Optional) Return information related to the specified class of objects.

`-message <arg>` - (Optional) Return information related to the specified message. Messages are specified in the form of a unique global message ID, that consists of an application sub-system code and a message identifier: "Common 17-24", or {Common 17-24}. Refer to the `set_msg_config` command for more information.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<pattern_or_object> - (Optional) Returns information related to the specified command, or a list of commands that match the specified pattern.

Note: A Vivado first class object, like a cell or site, must be specified when used with `-class` and `-prop` to return information about properties.

Examples

The following example returns a list of Xilinx Tcl command categories:

```
help
```

This example loads the simulator feature of the Vivado Design Suite, and then returns a list of Tcl commands in the simulation and waveform categories:

```
load_features simulator
help -category simulation
help -category waveform
```

Returns a list of all commands matching the specified search pattern:

```
help *file*
```

This list can be used to quickly locate a command for a specific purpose, such as `remove_files` or `delete_files`.

The following help command returns a long description of the `remove_files` command and its arguments:

```
help remove_files
```

Note: You can also use the `-args` option to get a brief description of the command.

This example defines a procedure called short, and returns the `-args` form of help for the specified command:

```
proc short cmdName {help -args $cmdName}
```

Note: You can add this procedure to your `init.tcl` file to load this command every time the tool is launched. Refer to *Chapter 1, Introduction* of the *Vivado Design Suite Tcl Command Reference (UG835)* for more information on the `init.tcl` file.

The following examples show how to obtain help for properties of design objects, or a class of design objects:

```
help -class cell -prop NAME
help -prop NAME [get_cells cpuEngine]
```

Note: In the preceding example, the first command returns general information related to the NAME property, while the second command also returns the value of the NAME property on the specified design object.

See Also

- get_cells
- list_features
- list_property
- load_features
- report_property
- set_msg_config

highlight_objects

Highlight objects in different colors

Syntax

```
highlight_objects [-color_index <arg>] [-rgb <args>] [-color <arg>]
     [-leaf_cells] [-quiet] [-verbose] <objects>
```

Usage

Name	Description
[-color_index]	Color index
[-rgb]	RGB color index list
[-color]	Valid values are red green blue magenta yellow cyan and orange
[-leaf_cells]	Leaf cells
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<objects>	Objects to highlight

Categories

GUIControl

Description

Highlights the specified or selected object or objects in a color as determined by one of the available color options.

> **TIP:** *Only one of the available color option should be used to specify the highlight color. However, if more than one color option is used, the order of precedence to define the color is* `-rgb`, `-color_index`, *and* `-color`.

Selected objects are automatically unselected in order to display the objects in the specified highlight color. Objects can be unhighlighted with the `unhighlight_objects` command.

Arguments

`-color_index <arg>` - (Optional) Valid values are integers from 1 to 20. Specifies the color index to use for highlighting the selected object or objects. The color indexes are displayed, and can be configured, on the **Colors → Highlight** page of the **Tools → Settings** dialog box. Refer to the *Vivado Design Suite User Guide: Using the IDE (UG893)* for more information on setting colors.

`-rgb` <args> - (Optional) The color to use in the form of an RGB code specified as {R G B}. For instance, {255 255 0} specifies the color yellow.

`-color` `<arg>` - (Optional) The color to use for highlighting the specified object or objects. Supported highlight colors are: red, green, blue, magenta, yellow, cyan, and orange.

Note: White is the color used to display selected objects with the `select_objects` command.

`-leaf_cells` `<arg>` - (Optional) Specifies one or more cell names or objects to highlight with the specified color.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<objects>` - (Required) Specifies one or more objects to be highlighted.

Examples

The following example highlights the currently selected objects in the color red:

```
highlight_objects -color red [get_selected_objects]
```

This example highlights the specified cells in green:

```
highlight_objects -color green -leaf_cells [get_cells cpuEngine/*]
```

See Also

- get_highlighted_objects
- get_marked_objects
- get_selected_objects
- mark_objects
- select_objects
- unhighlight_objects

Send Feedback

implement_debug_core

Implement debug core

Syntax

```
implement_debug_core [-quiet] [-verbose] [<cores>...]
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<cores>]	Debug core

Categories

Debug

Description

Implements the Vivado logic analyzer debug cores in the current project. The tools will be run once for any ILA debug cores specified, and run one more time for the Debug Hub core if all cores are specified. The ILA core (labtools_ila_v3) is the only core type currently supported by the `create_debug_core` command. The tool automatically adds a Debug Hub core (labtools_xsdbmasterlib_v2) to contain and configure the ILA cores in the project.

The Vivado tool creates Debug Hub core and ILA cores initially as black boxes. These cores must be implemented prior to running through place and route. After the core is created with `create_debug_core`, and while ports are being added and connected with `create_debug_port` and `connect_debug_port`, the content of the debug core is not defined or visible within the design.

Debug core implementation is automatic when you launch an implementation run using the `launch_runs` command, or during design optimization using `opt_design`. However, you can also use the `implement_debug_core` command to implement one or more of the cores in the design without having to implement the whole design.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<cores> - (Optional) One or more debug cores to implement. All debug cores will be implemented if no cores are specified.

Examples

The following example implements all debug cores in the current project:

```
implement_debug_core [get_debug_cores]
```

See Also

- connect_debug_port
- create_debug_core
- create_debug_port
- get_debug_cores
- launch_runs

implement_mig_cores

Call IP Services to regenerate an IP, then stitch it into the current netlist

Syntax

```
implement_mig_cores [-output_dir <arg>] [-rtl_only] [-force]
    [-debug_output] [-quiet] [-verbose]
```

Usage

Name	Description
[-output_dir]	Target Output Directory for PHY IP Generated Files Default: empty
[-rtl_only]	Run the complete process to generate the PHY RTL code but do not replace the PHY core netlist
[-force]	Implement all non-optimized memory cores. When use with -rtl_only, optimized cores will be included, as well.
[-debug_output]	Enable debugging output.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Memory

Description

Implements the memory IP cores in the current project.

Memory IP included in the Xilinx® IP catalog are used to generate memory controllers and interfaces for Xilinx devices. Memory IP includes different IP cores from the Xilinx IP catalog depending on the device architecture and memory interface specified. Refer to *Zynq-7000 SoC and 7 Series Devices Memory Interface Solutions (UG586)*, or *UltraScale Architecture-Based FPGAs Memory Interface Solutions (PG150)*, for details of the available memory IP.

The `implement_mig_cores` command generates the RTL information for the physical interface (PHY) of the memory controller, and integrates the synthesized netlist of the memory controller into the top-level design.

A memory controller can be debug enabled when added into the design from the Xilinx IP catalog. In the Vivado logic analyzer, or the Vivado Lab Edition, memory controllers implemented into a design are associated with hw_mig objects, one hw_mig object per debug-enabled memory controller. The hw_mig object will have all the properties needed to get the calibration status and draw the per-bit eye margin views.

Implementation of the memory IP, and debug core, is automatic when you launch an implementation run using the `launch_runs` command, or when you run `opt_design`. However, you can also use the `implement_mig_cores` command to integrate the memory IP without having to implement the whole design.

> **TIP:** *All pins of the memory controller must be assigned prior to running the* `implement_mig_cores` *command, or an error will be returned. You can use* `report_drc` *to check the status of the memory controller.*

This command returns a transcript of its process, or returns an error if it fails.

Arguments

`-outputdir <arg>` - (Optional) Specify the output directory for the generated output products of the memory IP. If `-outputdir` is not specified, the output will be written to the current project folders.

`-rtlonly` - (Optional) Generate only the PHY RTL information for the memory controller.

`-force` - (Optional) Force the implementation of the memory IP even if it is up-to-date.

`-debug_output` - (Optional) Enable the debugging feature of the memory IP.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example implements the memory IP cores in the current design:

```
implement_mig_cores
```

See Also

- commit_hw_mig
- get_hw_migs
- launch_runs
- opt_design
- refresh_hw_mig

- report_hw_mig

implement_xphy_cores

Call IP Services to regenerate an IP, then stitch it into the current netlist

Syntax

```
implement_xphy_cores [-output_dir <arg>] [-rtl_only] [-force]
    [-debug_output] [-update_delay_value_only] [-quiet] [-verbose]
```

Usage

Name	Description
[-output_dir]	Target Output Directory for PHY IP Generated Files Default: empty
[-rtl_only]	Run the complete process to generate the PHY RTL code but do not replace the PHY core netlist
[-force]	Implement all non-optimized memory cores. When use with -rtl_only, optimized cores will be included, as well.
[-debug_output]	Enable debugging output.
[-update_delay_value_only]	Update latest Delay value.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Memory

Implements the memory IP cores and Advanced IO Wizard cores in the current project that utilize XPHY Logic.

Memory IP included in the Xilinx IP catalog are used to generate memory controllers and interfaces for Xilinx devices. Memory IP includes different IP cores from the Xilinx IP catalog depending on the memory interface specified. More details can be found in the product guides for the individual Memory IP. For soft memory controllers, refer to the Versal ACAP RLDRAM 3 (PG354), QDR-IV (PG355), and DDR4 (PG353) product guides.

> 💡 **TIP:** *For hardened memory controllers, refer to the Versal ACAP Programmable Network on Chip and Integrated Memory Controller product guide (PG313).*

Advanced IO Wizard IP is included in the Xilinx IP catalog and are used to generate non-memory high-speed IO interfaces for Xilinx devices. For more information on the Advanced IO Wizard IP see the Advanced IO Wizard Product Guide (PG320).

The `implement_xphy_cores` command generates the RTL information for the physical interface (PHY) of the memory controller or Advanced IO Wizard interface, and integrates the synthesized netlist of the memory controller into the top-level design. Implementation of the memory IPs and Advanced IO Wizard interfaces is automatic when you launch an implementation run using the launch_runs command, or when you run `opt_design`. However, you can also use the `implement_xphy_cores` command to integrate the memory IP or Advanced IO Wizard interface without having to implement the whole design.

Note: All pins of the memory controller and Advanced IO Wizard interface must be assigned prior to running the implement_xphy_cores command, or an error will be returned. You can use `report_drc` to check the status of the memory controller. This command returns a transcript of its process, or returns an error if it fails.

Arguments

`-update_delay_value_only` – (Optional) Deposits the `DELAY_VALUE_XPHY` attribute value assigned to a port in your design to the proper `DELAY_VALUE_x` attribute on an XPHY instance for your Advanced IO Wizard interface. The `DELAY_VALUE_x` attribute on the XPHY interface represents the calibrated input delay or calibrated output delay value in picoseconds corresponding to a particular XPHY input pin `DATAIN[x]` or XPHY output pin `OO[x]`.

Valid values for the `DELAY_VALUE_XPHY` port attribute and `DELAY_VALUE_x` XPHY attribute range from 0 to 625 when the output delay is not used as a cascade. When an input pin uses a cascaded output delay, the valid values for the `DELAY_VALUE_XPHY` port attribute and `DELAY_VALUE_x` XPHY attribute range from 0 to 1250. This argument would typically be used in situations when you are tuning an interface in a lab environment on a fully implemented design.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example opens a fully implemented design checkpoint and then generates two separate device images with calibrated input delay values of 500 and 525 picoseconds to tune the interface. The `DELAY_VALUE_XPHY` is applied to the port object `Datain0_p` and then deposited onto the XPHY instance associated with the port using the `-update_delay_value_only` argument prior to generating the device images.

```
open_checkpoint advanced_io_wizard_example_routed.dcp

set_property DELAY_VALUE_XPHY 500 [get_ports Datain0_p]
implement_xphy_cores -update_delay_value_only
write_device_image Datain0_500ps.pdi

set_property DELAY_VALUE_XPHY 525 [get_ports Datain0_p]
write_device_image Datain0_525ps.pdi
```

Send Feedback

import_files

Import files and/or directories into the active fileset

Syntax

```
import_files [-fileset <arg>] [-force] [-of_objects <args>] [-norecurse]
    [-flat] [-relative_to <arg>] [-quiet] [-verbose] [<files>...]
```

Returns

A list of file objects that were imported.

Usage

Name	Description
[-fileset]	Fileset name
[-force]	Overwrite files of the same name in project directory
[-of_objects]	RMs to import the files to
[-norecurse]	Disables the default behavior of recursive directory searches
[-flat]	Import the files into a flat directory structure
[-relative_to]	Import the files with respect to the given relative directory
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<files>]	Name of the files to import into fileset

Categories

Project, Simulation

Description

Imports one or more files or the source file contents of one or more directories to the specified fileset.

For every file added to a project the Vivado Design Suite attempts to store and maintain both a relative path and an absolute path to the file or directory. When a project is opened, these paths are used to locate the files and directories. By default the Vivado Design Suite applies a Relative First approach to resolving paths, searching the relative path first, then the absolute path. You can use the PATH_MODE property to change how the Vivado tool resolves file paths or properties for specific objects. For more information, see the *Vivado Design Suite Properties Reference Guide (UG912)*.

> ⭐ **IMPORTANT!** *Importing multiple files one at a time can cause noticeable performance degradation. It is more efficient to use a single `import_files` command to import a list of files:*

```
import_files {file1 file2 file3 ... fileN}
```

This command is different from the `add_files` command, which adds files by reference into the specified fileset. This command imports the files into the local project folders under `project.srcs\<fileset>\imports` and then adds the file to the specified fileset.

Arguments

`-fileset <name>` - (Optional) The fileset to which the specified source files should be added. If the specified fileset does not exist, the tool will return an error. If no fileset is specified the files will be added to the source fileset by default.

`-force` - (Optional) Overwrite files of the same name in the local project directory and in the fileset.

`-norecurse` - (Optional) Do not recurse through subdirectories of any specified directories. Without this argument the tool will also search through any subdirectories for additional source files that can be added to a project.

`-flat` - (Optional) Import all files into the imports folder without preserving their relative paths. By default the directory structure of files is preserved as they are imported into the design.

`-relative_to <arg>` - (Optional) Import the files relative to the specified directory. This allows you to preserve the path to the imported files in the directory structure of the local project. The files will be imported to the imports folder with the path relative to the specified directory.

Note: The `-relative_to` argument is ignored if the `-flat` argument is also specified. The `-flat` command eliminates the directory structure of the imported files.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<files>` - (Optional) One or more file names or directory names to be added to the specified fileset. If a directory name is specified, all valid source files found in the directory, and in subdirectories of the directory, will be added. If no files are specified, the tool imports files in the source set for the current project.

Note: If the path is not specified as part of the file name, the current working directory is used, or the directory from which the tool was launched.

Examples

The following example imports the `top.ucf` file into the constrs_1 constraint fileset.

```
import_files -fileset constrs_1 top.ucf
```

The following example imports the valid source files into the source fileset (`sources_1`) as a default since the `-fileset` argument is not specified. In addition, the `-norecurse` argument restricts the tool to looking only in the specified `\level1` directory and not searching any subdirectories. All valid source files will be imported into the `\imports` folder of the project because the `-flat` argument has been specified.

```
import_files C:/Data/FPGA_Design/level1 -norecurse -flat
```

Note: Without the `-flat` option a `\level1` directory would be created inside of the `\imports` folder of the project.

The following example imports files into the source fileset (`sources_1`) because the `-fileset` argument is not specified. Valid source files are imported from the `\level1` directory, and all subdirectories, and the files will be written into the `\imports` folder of the project starting at the `\Data` directory due to the use of the `-relative_to` argument.

```
import_files C:/Data/FPGA_Design/level1 -relative_to C:/Data
```

See Also

- add_files

Send Feedback

import_ip

Import an IP file and add it to the fileset

Syntax

```
import_ip [-srcset <arg>] [-name <arg>] [-quiet] [-verbose] [<files>]
```

Returns

List of file objects that were added.

Usage

Name	Description
[-srcset]	(Optional) Specifies the source file set containing the objects to be upgraded Default: The current source fileset Values: Source set name
[-name]	(Optional) Specifies a replacement name for the imported IP; may not be used with multiple files. Default: The current name for the imported IP Values: The new name for the imported IP
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<files>]	Names of the IP files to be imported Values: A list of XCI (and/or XCO) file name(s)

Categories

Project, IPFlow

Description

Adds an existing XCI or XCO file as an IP source into the current project, and copies it into the local project directory structure.

The `import_ip` command allows you to read existing IP files directly, and copy them into the local project folders. Use the `read_ip` or `add_files` command to add IP files by reference into the current project.

Use the `create_ip` command to create new IP files from the current IP catalog.

Arguments

`-srcset <arg>` - (Optional) Specifies the source file set to import the IP files into. If not specified, the default source file set is `sources_1`.

Send Feedback

`-name <arg>` - (Optional) The name to assign to the IP object as it is added to the current source fileset. This option can only be used when a single file is specified in `<files>`.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<files>` - (Optional) The names of the IP files to be imported into the current project. Each IP must be in the form of an existing XCI file or XCO file. An XCI file is an IP-XACT format file that contains information about the IP parameterization. An XCO file is a CORE Generator log file that records all the customization parameters used to create the IP core and the project options in effect when the core was generated. The XCI or XCO files are used to recreate the core in the current project.

Note: If the path is not specified as part of the file name, the tool will search for the specified file in the current working directory and then in the directory from which the tool was launched.

Examples

The following example copies the 10gig ethernet core into the current project, and assigns it a name of IP_block1:

```
import_ip C:/Data/FPGA_Design/10gig_eth.xci -name IP_block1
```

See Also

- add_files
- create_ip
- generate_target
- read_ip

Send Feedback

import_synplify

Imports the given Synplify project file

Syntax

```
import_synplify [-copy_sources] [-quiet] [-verbose] <file>
```

Returns

List of files object that were imported from the Synplify file.

Usage

Name	Description
[-copy_sources]	Copy all the sources from synplify project file into the created project
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<file>	Name of the Synplify project file to be imported

Categories

Project

Description

Imports Synplify synthesis project files (`.prj`) into the current project, including the various source files used in the synthesis run.

Arguments

`-copy_sources` - (Optional) Copy Synplify project source files to the local project directory structure rather than referencing them from their current location. The default is to reference source files from their current location.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Send Feedback

`<file>` - (Required) The name of the Synplify project file from which to import the source files.

Examples

The following example creates a new project and imports the specified Synplify project file, copying the various source files from the Synplify project into the local project directories:

```
create_project syn_test C:/Data/FPGA_Design/syn_test
import_synplify -copy_sources C:/Data/syn_data.prj
```

See Also

- create_project

import_xise

Import XISE project file settings into the created project

Syntax

```
import_xise [-copy_sources] [-quiet] [-verbose] <file>
```

Returns

True

Usage

Name	Description
`[-copy_sources]`	Copy all ISE sources into the created project
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<file>`	Name of the XISE project file to be imported

Categories

Project

Description

Imports an ISE project file (XISE) into the current project. This allows ISE projects to be quickly migrated into the Vivado Design Suite for synthesis, simulation, and implementation. All project source files, constraint files, simulation files, and run settings are imported from the ISE project and recreated in the current project.

This command should be run on a new empty project. Since source files, constraints, and run settings are imported from the ISE project, any existing source files or constraints may be overwritten.

Arguments

`-copy_sources` - (Optional) Copy source files in the ISE project to the local project directory structure rather than referencing them from their current location. The default is to reference source files from their current location.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file>` - (Required) The name of the ISE project file (.XISE) to be imported into the current project.

Examples

The following example creates a new project called importISE, and then imports the ISE project file (`first_use.xise`) into the new project.

```
create_project importISE C:/Data/importISE import_xise \
C:/Data/FPGA_design/ise_designs/drp_des/first_use.xise
```

Note: This example does not specify the `-copy_sources` argument, so all source files in the ISE project will be added to the current project by reference.

See Also

- create_project

import_xst

Imports the given XST project file

Syntax

```
import_xst [-copy_sources] [-quiet] [-verbose] <file>
```

Returns

List of files object that were imported from the XST file.

Usage

Name	Description
[-copy_sources]	Copy all the sources from xst project file into the created project
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<file>	Name of the XST project file to be imported

Categories

[Project](#)

Description

Imports XST synthesis project files into the current project, including the various source files used in the XST run.

Arguments

`-copy_sources` - (Optional) Copy XST project source files to the local project directory structure rather than referencing them from their current location. The default is to reference source files from their current location.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Send Feedback

`<file>` - (Required) The name of the XST project file from which to import the source files.

Examples

The following example creates a new project called `xst_test`, and imports the `drp_des.xst` file:

```
create_project xst_test C:/Data/FPGA_Design/xst_test
import_xst C:/Data/ise_designs/drp_des.xst
```

See Also

- create_project

include_bd_addr_seg

include segment from an address space.

Syntax

```
include_bd_addr_seg [-quiet] [-verbose] [<segment_to_include>]
```

Returns

The newly included segment object, "" if failed.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<segment_to_include>]	Segment to include

Categories

IPIntegrator

Description

Reverses the exclusion of an AXI peripheral address segment from access by the AXI master, and restores the address segment to a mapped state.

In the block design, address segments of AXI peripherals can have one of three states:

- Unmapped - An AXI peripheral, or slave interface, is connected to an AXI master, but the peripheral has not been assigned an address segment in the master's address space and is not visible to the master.

- Mapped - The AXI peripheral is mapped into the AXI master's address space, assigned an address segment or range, and is accessible through the master.

- Excluded - The AXI peripheral is mapped to the AXI master, and has been assigned an address segment, but is not accessible to the master. The address segment that the AXI slave occupies within the master address space is also considered filled.

The `exclude_bd_addr_seg` command lets you exclude specific address segments from access by the AXI master they are mapped to. The `include_bd_addr_seg` restores access to the mapped address segment.

This command returns nothing if successful, or returns an error if it failed.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<segment_to_include> - A single address segment object, bd_addr_seg, to restore to the AXI master address space.

Example

The following example restores the AXI peripheral address segment for access by its AXI master:

```
include_bd_addr_seg [get_bd_addr_segs microblaze_1/Data/SEG_axi_gpio_1_Reg]
```

See Also

- assign_bd_address
- create_bd_addr_seg
- exclude_bd_addr_seg
- get_bd_addr_segs
- get_bd_addr_spaces

Send Feedback

infer_diff_pairs

Infer differential pairs, typically for ports just imported from a CSV or XDC file

Syntax

```
infer_diff_pairs [-file_type <arg>] [-quiet] [-verbose] [<file>...]
```

Usage

Name	Description
[-file_type]	Input file type: 'csv' or 'xdc' Default: file type
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<file>]	Pin Planning CSV or XDC file Default: file

Categories

[FileIO](#)

Description

The `infer_diff_pairs` command can be used in an I/O Pin Planning project, after importing the I/O pin information using the `read_csv` or `read_xdc` command.

There are several attributes that identify differential pairs in the file: Signal Name, DiffPair Signal, DiffPair Type, and I/O Standard.

The tool will identify differential pairs using the following methods:

- Matching Diff Pair - This is a direct definition of the two signals which make up a differential pair. Two port entries, each have DiffPair Signal values linking to the Signal Name of the other, and have complementary DiffPair Type values, one N and one P. The tool checks to ensure that the other attributes such as I/O Standard are compatible when forming the diff pair.

- Unmatched Diff Pair - Two port entries, with complementary DiffPair Type values (one N, one P), but only one port has a DiffPair Signal linking to the other Signal Name. The tool will create the differential pair if all other attributes are compatible.

- Single Port Diff Pair - A single port entry with a differential I/O Standard, a DiffPair Type value, and a DiffPair Signal that does not otherwise appear in the CSV. The tool will create the opposite side of the differential pair (the N or P side), with all properties matching those on the original port.

- Inferred Diff Pair - Two ports entries, with Signal Names that imply the N and P side. The tool will infer a differential pair if all other attributes are compatible.

After reading the port definitions from a CSV or XDC file, the tool will report that some differential pairs can be inferred from the data. You can run the `infer_diff_pairs` command to infer these differential pairs if you choose.

Arguments

`-file_type [ csv | xdc ]` - (Optional) Specify the type of file to import when inferring differential pairs. The valid file types are CSV and XDC. There is no default; the `-file_type` must be specified.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file>` - (Optional) The name of the file previously imported.

Note: If the path is not specified as part of the file name, the tool will search for the specified file in the current working directory and then in the directory from which the tool was launched.

Examples

The following example imports the specified XDC file, and then infers differential pairs from the file:

```
read_xdc C:/Vivado_Install/io_1.xdc
infer_diff_pairs C:/Vivado_Install/io_1.xdc -file_type xdc
```

See Also

- read_csv
- read_xdc

instantiate_example_design

Creates an example design from a predefined template in an open project.

Syntax

```
instantiate_example_design [-design <arg>] [-hier <arg>] [-project <arg>]
    [-project_location <arg>] [-options <args>] [-quiet] [-verbose]
    <template>
```

Returns

Returns the name of the template applied.

Usage

Name	Description
[-design]	Block Design Name
[-hier]	Hierarchy Block
[-project]	Project Name
[-project_location]	Project location Default: .
[-options]	Configurable options
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<template>	Configurable Design Name

Categories

IPIntegrator

Description

This command creates an example design from a predefined template in an open project. The target part specified by the open project must be compatible with the example design, as defined in the SUPPORTED_PARTS property of the example, or an error is returned.

For the embedded processor example designs, base_microblaze and base_zynq, the example design must be created in an open block design in the Vivado IP integrator. Embedded processor example designs require the use of a board as defined by the BOARD_PART property, rather than a target part. Refer to the `current_board_part` command for more information.

The command returns the name of the example design used and a transcript of commands; or it returns an error if it fails.

Send Feedback

Arguments

`-design <arg>` - (Optional) For embedded processor example designs, this option specifies the name of the open and current block design to instantiate the example design into. This option is required for embedded processor example designs. If `-design` is not specified, an error is returned.

`-hier` - (Optional) Hierarchy Block

`-options <args>` - (Optional) Specify the values of configurable properties of the example design.

> 💡 **TIP:** *The configurable properties (CONFIG.*) of an example design can be returned by the* `report_property` *or* `get_property` *commands.*

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<template>` - Specifies the template design to be instantiated into the specified design. The template can be specified by name, or as an object returned by the `get_example_designs` command.

Examples

The following example creates a new project as specified, overwriting an existing project of the same name if one is found, specifies the BOARD_PART property for the project, creates a new empty block design in the Vivado IP integrator, and then instantiates the Zynq embedded processor example design:

```
create_project zynq1 -force
set_property BOARD_PART em.avnet.com:zed:1.3 [current_project]
create_bd_design myFirstZynq
instantiate_example_design -design myFirstZynq \
[lindex [get_example_designs] 1]
```

This example reports the configurable properties of the specified example design:

```
report_property [lindex [get_example_designs] 3] CONFIG.*
```

This example creates a new empty project as specified, sets a target BOARD for the project, creates and opens a new empty block design, and then instantiates the configurable example design:

```
create_project mb1 C:/Data/Vivado_Tutorial/Tutorial_Created_Data/mb1
set_property board_part xilinx.com:kcu105:part0:1.1 [current_project]
create_bd_design design_1
instantiate_example_design -design design_1 \
    -options { Data_Cache.VALUE 8K Include_DDR4.VALUE true \
    Local_memory.VALUE 128K }\
xilinx.com:design:config_mb:1.0
```

See Also

- create_bd_design
- create_project
- get_example_designs
- set_property

instantiate_template_bd_design

Creates a block design in IP integrator from a predefined template.

Syntax

```
instantiate_template_bd_design -design <arg> [-hier <arg>]
    [-options <args>] [-quiet] [-verbose] <template>
```

Returns

Returns the name of the template applied.

Usage

Name	Description
`-design`	Block Design Name
`[-hier]`	Hierarchy Block
`[-options]`	Configurable options
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<template>`	Configurable Design Name

Categories

[IPIntegrator](#)

Description

This command creates an example design from a template Block Design in the IP integrator of the Vivado Design Suite.

The template diagram is created in an existing and open block design. In addition, the target part specified by the current project or in-memory project must be compatible with the template design or an error is returned.

The command returns a transcript of its process, or returns an error if it fails.

Arguments

`-design <arg>` - (Required) Specifies the name of the block design to instantiate the template diagram into. The specified block design must exist and be open in the IP integrator, or an error is returned.

`-hier` - (Optional) Hierarchy Block

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<template> - Specifies the template diagram to be instantiated into the specified design. The template can be specified by name, or as an object specified by the `get_template_db_designs` command.

Examples

The following example builds the specified template block design in the specified design:

```
instantiate_template_bd_design -design myFirstZynq \
[lindex [get_template_bd_designs] 1]
```

See Also

- get_template_bd_designs

iphys_opt_design

Interactive phys_opt_design.

Syntax

```
iphys_opt_design [-fanout_opt] [-critical_cell_opt] [-replicate_cell]
    [-placement_opt] [-rewire] [-forward_retime] [-backward_retime]
    [-net <arg>] -cluster <args> -place_cell <args> [-dsp_register_opt]
    [-bram_register_opt] [-uram_register_opt] [-shift_register_opt]
    [-cell <arg>] [-packing] [-unpacking] [-port <arg>] [-critical_pin_opt]
    [-restruct_opt] [-equ_drivers_opt] [-skipped_optimization]
    [-create_bufg] [-insert_negative_edge_ffs] [-hold_fix]
    [-slr_crossing_opt] [-shift_register_to_pipeline] [-auto_pipeline]
    [-pipeline_to_shift_register] [-quiet] [-verbose]
```

Usage

Name	Description
[-fanout_opt]	Fanout optimization including very high fanout optimizations
[-critical_cell_opt]	Do cell-duplication based optimization on timing critical nets
[-replicate_cell]	Atomic cell replication operation, for interconnect retiming and LUT optimizaton
[-placement_opt]	Move cells to reduce delay on timing-critical nets
[-rewire]	Do rewiring optimization
[-forward_retime]	Forward retiming optimization
[-backward_retime]	Backward retiming optimization
[-net]	net to be optimized
-cluster	Clusters of load pins
-place_cell	Place cell or cell connecting to pin to loc
[-dsp_register_opt]	DSP register optimization
[-bram_register_opt]	BRAM register optimization
[-uram_register_opt]	UltraRAM register optimization
[-shift_register_opt]	Shift register optimization
[-cell]	cell to be optimized
[-packing]	Packing in DSP/BRAM
[-unpacking]	Unpacking in DSP/BRAM
[-port]	Port in DSP/BRAM that is optimized
[-critical_pin_opt]	Pin Swap optimization
[-restruct_opt]	LUT restructuring optimization
[-equ_drivers_opt]	Equivalent driver rewiring
[-skipped_optimization]	The change is not committed
[-create_bufg]	Insert bufg to drive HFO nets
[-insert_negative_edge_ffs]	Inserting negative edge triggered FFs for high hold mitigation

Name	Description
`[-hold_fix]`	Inserting buffers for hold fix optimization
`[-slr_crossing_opt]`	Optimize slr crossing nets
`[-shift_register_to_pipeline]`	shift register to pipeline opt
`[-auto_pipeline]`	Auto pipeline
`[-pipeline_to_shift_register]`	pipeline_to_shift_register
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution

Categories

Tools

Description

The `iphys_opt_design` command describes a specific optimization that was performed by the `phys_opt_design` command, such as replicating a critical cell or pulling registers from a block RAM to improve critical path delay. The `iphys_opt_design` command includes all the information necessary to recreate both the post-optimization logical netlist and the placement changes required for the optimized netlist.

Interactive physical optimization can be used in two ways:

- Applying post-placement physical optimizations to the pre-placement netlist to improve the overall placement result and improve design performance.

- Saving the physical optimizations in a Tcl script to be repeated as needed.

The various optimizations performed by `phys_opt_design` can be written to an iphys_opt Tcl script by `write_iphys_opt_tcl`, and read into the design by the `read_iphys_opt_tcl` command.

> 💡 **TIP:** *The `iphys_opt_design` command is intended for use inside the iphys_opt Tcl script file. These commands can be edited in the context of the iphys_opt Tcl script, but they are not intended to be specified at the command line.*

This command returns a transcript of its processes, or an error if it fails.

Arguments

`-fanout_opt` - (Optional) Performs delay-driven optimization on the specified net, by replicating drivers to reduce delay.

`-critical_cell_opt` - (Optional) Replicate cells on specified nets to reduce delays.

`-placement_opt` - (Optional) Move cells to reduce delay on specified nets.

Send Feedback

`-rewire` - (Optional) Refactor logic cones to reduce logic levels and reduce delay on critical signals.

`-net <arg>` - (Optional) Specify the net to apply an optimization to.

`-cluster` <args> - (Required) Specify a cluster of load pins.

`-place_cell` <args> - (Required) Place the specified cells, or cells connected to the specified pins, on the device sites specified.

`-dsp_register_opt` - (Optional) Improve critical path delay by moving registers from slices to DSP blocks, or from DSP blocks to slices.

`-bram_register_opt` - (Optional) Improve critical path delay by moving registers from slices to block RAMs, or from block RAMs to slices.

`-uram_register_opt` - (Optional) Improve critical path delay by moving registers from slices to UltraRAMs, or from UltraRAMs to slices.

`-shift_register_opt` - (Optional) Perform shift register optimization to improve timing on negative slack paths between shift register cells (SRLs) and other logic cells.

`-cell <arg>` - (Optional) Specify a cell to apply an optimization to.

`-packing` - (Optional) Packing in DSP/BRAM.

`-unpacking` - (Optional) Unpacking in DSP/BRAM.

`-port <arg>` - (Optional) Specify a port on a cell to apply the optimization to.

`-critical_pin_opt` - For LUT inputs, this optimization performs remapping of logical pins to physical pins, also known as pin-swapping, to improve critical path timing.

`-skipped_optimization` - (Optional) Defines the specified optimization as not performed. These are optimizations identified by phys_opt_design that are skipped because suitable locations for optimized logic cannot be found. For example, BRAM register optimizations to improve slack that are skipped because no suitable locations can be found for the registers.

`-insert_negative_edge_ffs` - (Optional) Insert negative edge, or falling edge triggered flip flops to help manage hold timing.

`-hold_fix` - (Optional) Performs optimizations to insert data path delay buffers for hold fix optimization.

`-slr_crossing_opt` - (Optional) Performs post-place or post-route optimizations to improve the path delay of inter-SLR connections. The optimization adjusts the locations of the driver, load, or both along the SLR crossing after potential replication. For use with UltraScale and UltraScale+ devices.

`-auto_pipeline` - (Optional) Auto pipeline the design.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example performs a critical cell optimization on the specified net and cluster of ports:

```
iphys_opt_design -critical_cell_opt -net \
    {ADUR_CORE_INST/CPE_INST/CPE_ANT_RESOURCE_TDM_INST0 \
    /CPE_ANT_LINE_IQ_TDM_ANT0_INST/CPE_PN_MULT_INST/CPE_PN_MUL_INST3 \
    /Q_PNI_MULT_INST/pn_mult_reg[3][0]} \
    -cluster {pn_mult[3]_i_14_replica  {\
    {ADUR_CORE_INST/CPE_INST/CPE_ANT_RESOURCE_TDM_INST0 \
    /CPE_ANT_LINE_IQ_TDM_ANT0_INST/CPE_PN_MULT_INST/CPE_PN_MUL_INST2 \
    /Q_ADD_INST/pn_mult_reg[3]_i_6_CARRY8/S[0]}}}\
    -cluster {pn_mult[3]_i_14_replica_1  {\
    {ADUR_CORE_INST/CPE_INST/CPE_ANT_RESOURCE_TDM_INST0 \
    /CPE_ANT_LINE_IQ_TDM_ANT0_INST/CPE_PN_MULT_INST/CPE_PN_MUL_INST0 \
    /Q_ADD_INST/pn_mult_reg[3]_i_10_CARRY8/S[0]}}}\
```

The following example performs a shift register optimization on the specified cell:

```
iphys_opt_design -shift_register_opt -cell \
    {ADUR_CORE_INST/EMIF_INTERFACE_INST/EMIF_HOST_IF_INST/DLY_INST1 \
    /PD_INST_FPGA/delay_chain_reg[9][16]_srl9} -port D
```

See Also

- phys_opt_design
- read_iphys_opt_tcl
- report_phys_opt
- write_iphys_opt_tcl

launch_chipscope_analyzer

Issues an error that you can not run this command

Syntax

```
launch_chipscope_analyzer [-run <arg>] [-csproject <arg>] [-quiet]
    [-verbose]
```

Usage

Name	Description
[-run]	Implemented run to launch ChipScope Analyzer with
[-csproject]	ChipScope project
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

ToolLaunch

Description

Launches the ChipScope™ Pro Analyzer tool for the active run, or a specified Implemented Design run. You can setup a Netlist Design for use with ChipScope prior to implementation, using the `create_debug_core`, `create_debug_port`, and `connect_debug_port` commands.

The Implemented Design must also have a bitstream file generated by BitGen for `launch_chipscope_analyzer` to run. If BitGen has not been run, an error will be returned.

Note: It is not enough to use the `write_bitstream` command to create a bitstream file. You must follow the steps outlined below in the second example.

Arguments

`-run <arg>` - The run name to use when launching the ChipScope Pro Analyzer. The specified run must be implemented and have a bitstream (.bit) file generated. ChipScope will use the bitstream file and the `debug_nets.cdc` file from the specified run.

`-csproject <arg>` - The name of the project to open in ChipScope Pro Analyzer. If you do not specify the project name, the default project name of `csdefaultproj.cpj` will be used. When you specify the project name, you should also specify the `.cpj` extension.

Note: The project is created in the `project/project.data/sources_1/cs` folder.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example launches ChipScope Pro Analyzer, specifying the implementation run to use and the name of the ChipScope project to create:

```
launch_chipscope_analyzer -run impl_3 -csproject impl_3_cs_project
```

The following example sets the `add_step Bitgen` property for the impl_4 run, launches the impl_4 run, and then launches the ChipScope Pro Analyzer on the specified run:

```
set_property add_step Bitgen [get_runs impl_4]
launch_runs impl_4 -jobs 2
launch_chipscope_analyzer -run impl_4
```

Note: In this example the ChipScope project will be called `csdefaultproj.cpj`.

See Also

- connect_debug_port
- create_debug_core
- create_debug_port
- launch_runs
- set_property
- write_bitstream

launch_impact

Issues an error that you can not run this command

Syntax

```
launch_impact [-run <arg>] [-ipf <arg>] [-quiet] [-verbose]
```

Usage

Name	Description
[-run]	Implemented run to launch iMPACT with
[-ipf]	Project for iMPACT
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

ToolLaunch

Description

Launch iMPACT to configure your device and generate programming files. You can also read back and verify design configuration data, debug configuration problems, or execute XSVF files.

You must generate the bitstream file using `write_bitstream` prior to using iMPACT.

The command returns the list of files read.

Arguments

`-run` - (Optional) Launch iMPACT with the specified run. If no run is specified, then iMPACT is launched with the active implementation run.

`-ipf` - (Optional) Specify the iMPACT project file to use to save the results to. The iMPACT Project File (IPF) contains information from a previous session of iMPACT. The target device is configured according to the settings in the specified IPF file. If you do not specify `-ipf`, the target device is configured according to the default settings.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example launches iMPACT using the specified implementation run:

```
launch_impact -run impl_3
```

See Also

- write_bitstream

Send Feedback

launch_runs

Launch a set of runs

Syntax

```
launch_runs [-jobs <arg>] [-scripts_only] [-lsf <arg>] [-sge <arg>]
    [-cluster_configuration <arg>] [-dir <arg>] [-to_step <arg>]
    [-next_step] [-host <args>] [-remote_cmd <arg>] [-email_to <args>]
    [-email_all] [-pre_launch_script <arg>] [-post_launch_script <arg>]
    [-custom_script <arg>] [-force] [-quiet] [-verbose] <runs>...
```

Usage

Name	Description
[-jobs]	Number of jobs Default: 1
[-scripts_only]	Only generate scripts
[-lsf]	Use LSF to launch jobs. Required argument is the bsub command line to pass to LSF Default: empty
[-sge]	Use SGE to launch jobs. Required argument is the qsub command line to pass to SGE Default: empty
[-cluster_configuration]	Run jobs on a particular cluster configuration (picks cluster configuration from GUI settings). Default: empty
[-dir]	Launch directory
[-to_step]	Last Step to run. Ignored when launching multiple runs. Not valid with -next_step
[-next_step]	Run next step. Ignored when launching multiple runs. Not valid with -to_step.
[-host]	Launch on specified remote host with a specified number of jobs. Example: -host {machine1 2} -host {machine2 4}
[-remote_cmd]	Command to log in to remote hosts Default: ssh -q -o ConnectTimeout=30 -o ConnectionAttempts=3 -o BatchMode=yes
[-email_to]	List of email addresses to notify when jobs complete (only applicable with -host)
[-email_all]	Send email after each job completes (only applicable with -host)
[-pre_launch_script]	Script to run before launching each job (only applicable with -host)
[-post_launch_script]	Script to run after each job completes (only applicable with -host)
[-custom_script]	User run script map file which contains run name to user run script mapping
[-force]	Run the command, even if there are pending constraint changes, which will be lost (in a Dynamic Function eXchange design)
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<runs>	Runs to launch

Send Feedback

Categories

[Project](#)

Description

Launches synthesis and implementation runs when running the Vivado tools in Project Mode. Refer to the *Vivado Design Suite User Guide: Design Flows Overview* (UG892) for a complete description of Project Mode and Non-Project Mode.

A run must be previously defined using the `create_run` command, and the properties of the run must be previously configured using the `set_property` command. Both synthesis and implementation runs can be specified in the same `launch_runs` command. However, to launch an implementation run, the parent synthesis run must already be complete.

In Non-Project Mode, Vivado synthesis can be launched directly using the `synth_design` command, and does not require the use of a defined run.

In Non-Project Mode, Vivado implementation steps can be launched individually with the `opt_design`, `power_opt_design`, `place_design`, `route_design`, `phys_opt_design`, and `write_bitstream` commands.

Arguments

`-jobs <arg>` - (Optional) The number of parallel jobs to run on the local host. The number of jobs for a remote host is specified as part of the `-host` argument. You do not need to specify both `-jobs` and `-host`.

`-scripts_only` - (Optional) Generate a script called `runme.bat` for each specified run so you can queue the runs to be launched at a later time.

`-lsf <arg>` - (Optional) Use IBM Platform Load Sharing Facility (LSF) to launch synthesis and implementation runs. The bsub command line to pass to LSF must be specified as a required argument.

`-sge <arg>` - (Optional) Use Oracle Grid Engine or Sun Grid Engine (SGE) to launch synthesis and implementation runs. The `qsub` command line is a required argument to submit a job to SGE.

`-cluster_configuration <arg>` - (Optional) Run jobs on a particular cluster configuration. Name of the cluster configuration is specified to submit a job on a particular cluster.

- To get the list of the cluster configurations, use the `get_cluster_configurations` command.

- To create a cluster configuration, use the `create_cluster_configuration` command.

- To remove a cluster configuration, use the `remove_cluster_configurations` command.

- To validate a cluster configuration, use the `validate_cluster_configurations` command.

`-dir <arg>` - (Optional) The directory for the tool to write run results into. A separate folder for each run is created under the specified directory. As a default the tool will write the results of each run into a separate folder under the `<project>.runs` directory.

`-to_step <arg>` - (Optional) Launch the run through the specified step in the implementation process, and then stop. For instance, run implementation through the `place_design` step, and then stop. This will allow you to look at specific stages of a run without completing the entire run. The following are the valid steps for implementation runs.

- opt_design - Optionally optimize the logical design to more efficiently use the target device resources. This step is usually enabled by default even though it is an optional step.

- power_opt_design - Optionally optimize elements of the logic design to reduce power demands of the implemented FPGA.

- place_design - Place logic cells onto the target device. This is a required step.

- power_opt_design (Post-Place) - Optionally optimize power demands of the placed logic elements. This step must be enclosed in quotes or braces since it includes multiple words (e.g. `-to_step "power_opt_design (Post-Place)"`).

- phys_opt_design - Optionally optimize design timing by replicating drives of high-fanout nets to better distribute the loads.

- route_design - Route the connections of the design onto the target FPGA. This is a required step.

- write_bitstream - Generate a bitstream file for Xilinx device configuration. This is a required step.

Note: The specified `-to_step` must be enabled for the implementation run using the `set_property` command, or the Vivado tool will return an error.

`-next_step` - (Optional) Continue a prior run from the step at which it was stopped. This option can be used to complete a run previously launched with the `-to_step` argument.

Note: The `-to_step` and `-next_step` arguments may not be specified together, and are ignored when launching multiple runs.

`-host <args>` - (Optional) Launch on the named remote host with a specified number of jobs. The argument is in the form of {<hostname> <jobs> }, for example: `-host {machine1 2}`. If the `-host` argument is not specified, the runs will be launched from the local host.

Note: This argument is supported on the Linux platform only.

`-remote_cmd <arg>` - (Optional) The command to use to login to the remote host to launch jobs. The default remote command is `"ssh -q -o BatchMode=yes"`.

`-email_to` **<args>** - (Optional) Email addresses to send a notification to when the runs have completed processing. This option also requires the use of `-host` with an SMTP server running to send Email notifications.

`-email_all` - (Optional) Send a separate Email for each run as it completes. This option also requires the use of `-host` with an SMTP server running to send Email notifications.

`-pre_launch_script <arg>` - (Optional) A shell script to run on the specified host before launching each job. This option also requires the use of `-host`.

`-post_launch_script <arg>` - (Optional) A shell script to run on the specified host after completion of all jobs. This option also requires the use of `-host`.

`-custom_script <arg>` - (Optional) Specify a script map file, which contains a mapping of run names and scripts to launch for that run.

`-force` - (Optional) Launch the run regardless of any pending constraint changes for Partial Reconfiguration designs.

Note: This argument applies only to Partial Reconfiguration projects. Any pending constraint changes will be lost to the specified runs.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<runs> - (Required) The names of synthesis and implementation runs to launch. One or more run names may be specified.

Examples

The following command launches three different synthesis runs with two parallel jobs:

```
launch_runs synth_1 synth_2 synth_4 -jobs 2
```

Note: The results for each run will be written to a separate folder `synth_1`, `synth_2`, and `synth_4` inside of the **<project>** `.runs` directory.

The following example creates a results directory to write run results. In this case a separate folder named `impl_3`, `impl_4`, and `synth_3` will be written to the specified directory. In addition, the `-scripts_only` argument tells the tool to write `runme.bat` scripts to each of these folders but not to launch the runs at this time.

```
launch_runs impl_3 impl_4 synth_3 -dir C:/Data/FPGA_Design/results -
scripts_only
```

The following example configures the impl_1 run, setting options for Vivado Implementation 2013, enabling some of the optional optimizations, and then launches the run to the `place_design` step:

```
set_property flow {Vivado Implementation 2013} [get_runs impl_1]
set_property STEPS.POWER_OPT_DESIGN.IS_ENABLED true [get_runs impl_1]
set_property STEPS.POST_PLACE_POWER_OPT_DESIGN.IS_ENABLED true \
    [get_runs impl_1]
set_property STEPS.PHYS_OPT_DESIGN.IS_ENABLED true [get_runs impl_1]
launch_runs -to_step place_design impl_1
```

See Also

- create_run
- get_runs
- opt_design
- phys_opt_design
- place_design
- power_opt_design
- reset_runs
- route_design
- set_property
- synth_design
- write_bitstream

Send Feedback

launch_simulation

Generate simulation scripts of the design and launch steps for the target simulator.

Syntax

```
launch_simulation [-step <arg>] [-simset <arg>] [-mode <arg>] [-type <arg>]
    [-scripts_only] [-of_objects <args>] [-absolute_path]
    [-install_path <arg>] [-gcc_install_path <arg>] [-noclean_dir] [-quiet]
    [-verbose]
```

Usage

Name	Description
[-step]	Launch a simulation step. Values: all, compile, elaborate, simulate. Default:all (launch all steps). Default: all
[-simset]	Name of the simulation fileset
[-mode]	Simulation mode. Values: behavioral, post-synthesis, post-implementation Default: behavioral
[-type]	Netlist type. Values: functional, timing. This is only applicable when mode is set to post-synthesis or post-implementation
[-scripts_only]	Only generate scripts
[-of_objects]	Generate compile order file for this object (applicable with -scripts_only option only)
[-absolute_path]	Make design source file paths in 'absolute' format
[-install_path]	Custom installation directory path
[-gcc_install_path]	Specify GNU compiler installation directory path for the g++/gcc executables
[-noclean_dir]	Do not remove simulation run directory files
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[ToolLaunch](#), [Simulation](#)

Description

Launch a simulator to perform analysis and verification of a design.

The `launch_simulation` command creates a script file for the target simulator and then executes this file in the simulation run directory. The simulation results are saved in the log files created in the run directory.

To run simulation for a specific simulator, you must first define the target simulator by setting the `TARGET_SIMULATOR` property on the design project:

```
set_property TARGET_SIMULATOR <name> [current_project]
```

The `TARGET_SIMULATOR` property can have a value of `XSim`, `ModelSim`, `IES`, `Xcelium`, `VCS`, `Riviera`, or `ActiveHDL`. The default value is `XSim`, the Vivado simulator.

The target simulator can also be defined from the Vivado IDE. Create or open a project, select **Tools → Settings → Simulation** menu item, and select the Target simulator from the drop-down menu. The available choices are: Vivado simulator, ModelSim Simulator, Questa Advanced Simulator, Incisive Enterprise Simulator (IES), Xcelium Parallel Simulator, Verilog Compiler Simulator (VCS), Riviera-PRO Simulator, and Active-HDL Simulator.

> **TIP:** *Some of these simulators are only available on Linux and some are only available on Windows.*

The `launch_simulation` command uses a three-step process comprised of compile, elaborate, and simulate steps. A script file for the target simulator is created for each step in the process, (`compile.bat`, `elaborate.bat`, `simulate.bat`), and written to the simulation run directory.

> **TIP:** *On Linux the script files are named with the `.sh` suffix instead of `.bat`.*

By default, `launch_simulation` will run these script files in sequence to run the simulation. You can create the scripts without running them by using the `-scripts_only` option.

This command returns a transcript of its process, or returns an error if it fails.

Arguments

`-simset <arg>` - (Optional) The name of the simulation fileset containing the simulation test benches and sources to be used during simulation. If not specified, the current simulation fileset is used.

`-mode [ behavioral | post-synthesis | post-implementation ]` - (Optional) Specifies either a behavioral simulation of the HDL design sources to verify syntax and confirm that the design performs as intended, a functional or timing simulation of the post-synthesis netlist, or a functional or timing simulation of the post implementation design to verify circuit operation after place and route. The default mode is behavioral.

`-type [ functional | timing ]` - (Optional) Specifies functional simulation of just the netlist, or timing simulation of the netlist and SDF file. This option must be specified with `-mode` for post-synthesis or post-implementation, but cannot be used with `-mode behavioral`. Post-synthesis timing simulation uses SDF component delays from the `synth_design` command. Post-implementation timing simulation uses SDF delays from the `place_design` and `route_design` commands.

> ⭐ **IMPORTANT!** *Do not use `-type` with `-mode behavioral`, or the tool will return an error.*

`-scripts_only` - (Optional) Only generate the simulation scripts for the target simulator, rather than actually launching these scripts to start the "compile", "elaborate" and "simulate" steps. You can use the scripts to launch the simulation flow at a later time.

`-of_objects <arg>` - (Optional) Run simulation for a single specified sub-design, or composite file. The sub-design must be specified as a design object as returned by the `get_files` command, rather than simply specified by name.

`-absolute_path` - (Optional) Specify this option to define the source and include paths used in the simulation scripts as absolute paths. By default, all paths are written as relative to the simulation run directory. Relative paths include an "`origin_dir`" variable that is set in the simulation script to the current run directory, but you can edit the `$origin_dir` variable to point to a path of your choice when relocating the design and simulation scripts.

`-install_path <arg>` - (Optional) Specifies the directory containing simulator executables (e.g. `vlog.exe`, `ncvlog`, `vlogan`). If this option is not specified, the target simulator will be looked for in the current `$PATH`.

`-gcc_install_path <arg>` - (Optional) Specifies the directory containing simulator GCC executables (e.g. `gcc`, `g++`). If this switch is not specified, then the directory path will be determined from the project property named "`simulator.<name>_gcc_install_dir`". If this property is not set, then the path set by the `GCC_SIM_EXE_PATH` environment variable will be used. If this variable is not set, then launch_simulation will automatically try to calculate the path from the simulator installation. If not able to determine, the path will then be searched from the `$PATH` (Linux) or `%PATH%` (Windows) environment variable.

Supported GCC versions.

- questa - 5.3.0

- xcelium - 6.3

- vcs - 6.2.0

`-noclean_dir` - (Optional) Do not remove files from the simulation run directory prior to launching the simulator. The default behavior is to remove files from the simulation run directory to create a clean start. With the `-noclean_dir` option, existing files in the run directory are left in place. However, some of the files generated for use by the simulator will be overwritten or updated by re-launching the simulator.

`-step` - (Optional) Specifies the simulation step to be launched. The valid steps are: Compile, Elaborate, Simulate.

> 💡 **TIP:** *By default, all the steps will be executed.*

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following commands run behavioral simulation of the design using the Vivado simulator:

```
set_property target_simulator "XSim" [current_project]
launch_simulation
```

The following commands run post-synthesis functional simulation of the design using the ModelSim Simulator:

```
set_property target_simulator "ModelSim" [current_project]
launch_simulation -mode "post-synthesis" -type "functional"
```

The following commands run post-implementation functional simulation of the design using the Cadence IES Simulator:

```
set_property target_simulator "IES" [current_project]
launch_simulation -mode "post-implementation" -type "functional"
```

The following commands run post-implementation timing simulation of the design using the Synopsys VCS Simulator:

```
set_property target_simulator "VCS" [current_project]
launch_simulation -mode "post-implementation" -type "timing"
```

Send Feedback

The following command generates behavioral simulation scripts for the target simulator in the simulation run directory:

```
launch_simulation -scripts_only
```

The following commands run behavioral simulation flow of the design for the "my_simset" simulation fileset for the target simulator in the simulation run directory:

```
launch_simulation -simset [get_filesets my_simset]
```

The following command runs behavioral simulation flow for the `char_fifo.xci` IP for the target simulator in the simulation run directory, and does not clean up prior simulation files:

```
launch_simulation -noclean_dir -of_objects [get_files char_fifo.xci]
```

The following command generates absolute paths for the source files in the generated script files:

```
launch_simulation -absolute_path
```

The following command will pick the simulator tools from the specified installation path instead of from the PATH variable:

```
launch_simulation -install_path /tools/ius/13.20.005/tools/bin
```

See Also

- close_sim
- current_sim
- relaunch_sim
- xsim

limit_vcd

Limit the maximum size of the VCD file on disk (equivalent of $dumplimit verilog task)

Syntax

```
limit_vcd [-quiet] [-verbose] <filesize>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<filesize>	Specify the maximum size of the VCD file in bytes.

Categories

Simulation

Description

Specify the size limit, in bytes, of the Value Change Dump (VCD) file. This command operates like the Verilog `$dumplimit` simulator directive.

When the specified file size limit has been reached, the dump process stops, and a comment is inserted into the VCD file to indicate that the file size limit has been reached.

Note: You must run the `open_vcd` command before using the `limit_vcd` command.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<filesize> - (Required) Specify the file size limit of the open VCD file in bytes.

Examples

The following example limits the current VCD file:

```
limit_vcd 1000
```

See Also

- checkpoint_vcd
- flush_vcd
- log_vcd
- open_vcd

link_design

Open a netlist design

Syntax

```
link_design [-name <arg>] [-part <arg>] [-constrset <arg>] [-top <arg>]
    [-mode <arg>] [-pr_config <arg>] [-reconfig_partitions <args>]
    [-partitions <args>] [-ignore_timing] [-quiet] [-verbose]
```

Returns

Design object

Usage

Name	Description
[-name]	Design name
[-part]	Target part
[-constrset]	Constraint fileset to use
[-top]	Specify the top module name when the structural netlist is Verilog
[-mode]	The design mode. Values: default, out_of_context Default: default
[-pr_config]	PR Configuration to apply while opening the design
[-reconfig_partitions]	List of reconfigurable partitions to load while opening the design
[-partitions]	List of partitions to load while opening the design
[-ignore_timing]	open a netlist design without the timing constraints.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Tools

Description

Opens a new or existing netlist design, linking the netlist files and constraints with the target part to create the design. This command is intended for use with netlist source files, such as files generated by third party synthesis tools, or Vivado synthesis through the `synth_design` command.

The DESIGN_MODE property for the current source fileset must be defined as GateLvl in order to open a netlist design. If not, you will get the following error:

```
ERROR: The design mode of 'sources_1' must be GateLvl.
```

The `-top` switch is required for third-party synthesis designs. The netlist for the design must be rooted in a specific module. For project-based designs you can specify the TOP property on the project. However, in non-project mode, you must use the `-top` option for the `link_design` command.

For project based designs with RTL source files, use `launch_runs` to launch synthesis or implementation, and then use the `open_run` command to open the design.

For non-project based designs, use the `open_checkpoint` command to open a checkpoint into memory, opening the design in Non-Project Mode. Refer to the *Vivado Design Suite User Guide: Design Flows Overview* (UG892) for more information on Project Mode and Non-Project Mode.

Arguments

`-name <arg>` - (Optional) This is the name assigned to the netlist design when it is opened by the Vivado tool. This name is for reference purposes, and has nothing to do with the top-level of the design or any logic contained within.

`-part <arg>` - (Optional) The Xilinx device to use when creating a new design. If the part is not specified the default part will be used.

`-constrset <arg>` - (Optional) The name of the constraint fileset to use when opening the design.

Note: The `-constrset` argument must refer to a constraint fileset that exists. It cannot be used to create a new fileset. Use `create_fileset` for that purpose.

`-top <arg>` - (Optional) The name of the top module of the design hierarchy of the netlist.

> ⭐ **IMPORTANT!** *When specifying `-top` with an EDIF netlist-based design, or design checkpoint (DCP) file, the name of the top-level cell must match the top-level cell defined in the EDIF or DCP file.*

`-mode [ default | out_of_context ]` - (Optional) If you have synthesized a block, and disabled IO buffer insertion, you can load the resulting EDIF into the Vivado Design Suite using `-mode out_of_context`. This enables implementation of the module without IO buffers, prevents optimization due to unconnected inputs or outputs, and adjusts DRC rules appropriately for the design. Refer to the *Vivado Design Suite User Guide: Hierarchical Design (UG905)* for more information.

`-pr_config <arg>` - (Optional) For the Partial Reconfiguration (PR) project-based design flow, this option specifies the PR Configuration to apply while opening the design. For PR designs, the `create_pr_configuration` command lets you associate a Reconfigurable Module (RM) with each Partition Definition in the design. This option tells the Vivado tool to link the design checkpoint files for the RMs into the design. See the *Vivado Design Suite User Guide: Dynamic Function eXchange* (UG909) for more information.

`-reconfig_partitions <args>` - (Optional) Specify a list of reconfigurable partitions to load while opening the design. The specified reconfigurable partitions are marked with the HD.RECONFIGURABLE property for proper handling in the design.

`-partitions <args>` - (Optional) List of hierarchical design partitions to load while opening the design. Hierarchical design partitions are marked with the HD.PARTITION property. See the *Vivado Design Suite User Guide: Hierarchical Design* (UG905) for more information.

`-ignore_timing` - (Optional) open a netlist design without the timing constraints. This can be used to quickly open a design for evaluation, separate from the timing constraints.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following creates a new netlist design called Net1:

```
link_design -name Net1
```

Note: The default source set, constraint set, and part will be used in this example.

The following example opens a netlist design called Net1, and specifies the constraint set to be used:

```
link_design -name Net1 -constrset con1
```

See Also

- launch_runs
- open_checkpoint
- open_run

- synth_design

Send Feedback

list_features

List available features.

Syntax

```
list_features [-quiet] [-verbose]
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Tools

Description

In order to reduce the memory footprint of the Vivado Design Suite, there are groups of Tcl commands called "features" which are unavailable for use until you run a command from that feature set, or unless you explicitly load the feature using the `load_features` command.

This command lists the available features sets of the Vivado Design Suite that can be loaded with the `load_features` command.

Note: If a feature has been previously loaded, it will not be listed as a feature available to load.

This command returns a list of features, or an error message.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Send Feedback

Examples

The following example returns the list of features available to load into the Vivado Design Suite:

```
list_features
```

See Also

- help
- load_features

list_hw_samples

Return probe sample values

Syntax

```
list_hw_samples [-quiet] [-verbose] [<hw_probe>]
```

Returns

Samples

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<hw_probe>]	hw_probe object

Categories

Hardware, Object

Description

Writes data samples from the specified hw_probe object on the current hw_ila.

The number of captured samples returned from the specified probe is equal to the DATA_DEPTH property of the ILA core. The default data depth is 1024 samples. Data values are returned in the radix specified for the hw_probe, as determined by the DISPLAY_RADIX property.

> **TIP:** *For any samples to be returned, data must have been captured by the specified port.*

The values are listed to the standard output, or can be captured to a Tcl variable for post-processing, or output to a file.

The following is an example Tcl script that lists the data samples from hw_probes of interest:

```
# Define a list of probes to get the data samples from
set probeList [get_hw_probes *AR*]
#Specify the radix for the return values
set_property DISPLAY_RADIX BINARY [get_hw_probes *AR*]
# Define a filename to write data to
set fileName C:/Data/probeData1.txt
# Open the specified file in write mode
set FH [open $fileName w]
# Write probe data for each probe
```

Send Feedback

```
foreach x $probeList {
    puts $FH "$x:"
    puts $FH [list_hw_samples $x]
}
# Close the output file
close $FH
puts "Probe data written to $fileName\n"
```

This command returns the requested output, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_probe> - List the data samples for the specified hw_probe on the current hw_ila. The probe must be specified as an object returned by the `get_hw_probes` command.

Examples

The following example returns the data samples for the specified probe:

```
list_hw_samples [get_hw_probes *probe18]
```

See Also

- current_hw_ila
- get_hw_ilas
- get_hw_probes
- create_hw_probe

list_param

Get all parameter names

Syntax

```
list_param [-quiet] [-verbose]
```

Returns

List

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[PropertyAndParameter](#)

Description

Gets a list of user-definable configuration parameters. These parameters configure a variety of settings and behaviors of the tool. For more information on a specific parameter use the `report_param` command, which returns a description of the parameter as well as its current value.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Send Feedback

Examples

The following example returns a list of all user-definable parameters:

```
list_param
```

See Also

- get_param
- report_param
- reset_param
- set_param

list_property

List properties of object

Syntax

```
list_property [-class <arg>] [-regexp] [-quiet] [-verbose] [<object>]
    [<pattern>]
```

Returns

List of property names.

Usage

Name	Description
`[-class]`	Object type to query for properties. Ignored if object is specified.
`[-regexp]`	Pattern is treated as a regular expression
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`[<object>]`	Object to query for properties
`[<pattern>]`	Pattern to match properties against Default: *

Categories

Object, PropertyAndParameter

Description

Gets a list of all properties on a specified object or class.

Note: `report_property` also returns a list of properties on an object or class of objects, but also reports the property type and property value.

Arguments

`-class <arg>` - (Optional) Return the properties of the specified class instead of a specific object. The class argument is case sensitive, and most class names are lower case.

Note: `-class` cannot be used together with an `<object>`.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<object>` - (Optional) A single object on which to report properties.

Note: If you specify multiple objects you will get an error.

`<pattern>` - (Optional) Match the available properties on the `<object>` or `-class` against the specified search pattern. The `<pattern>` applies to the property name, and only properties matching the specified pattern will be reported. The default pattern is the wildcard '*' which returns a list of all properties on the specified object.

Note: The search pattern is case sensitive, and most properties are UPPER case.

Examples

The following example returns all properties of the specified CELL object:

```
list_property [get_cells cpuEngine]
```

The following example returns the properties matching the specified search pattern from the BEL class of objects:

```
list_property -class bel *NUM*
```

See Also

- create_property
- get_cells
- get_property
- list_property_value
- report_property
- reset_property
- set_property

list_property_value

List legal property values of object

Syntax

```
list_property_value [-default] [-class <arg>] [-quiet] [-verbose] <name>
    [<object>]
```

Returns

List of property values.

Usage

Name	Description
[-default]	Show only the default value.
[-class]	Object type to query for legal property values. Ignored if object is specified.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<name>	Name of property whose legal values is to be retrieved
[<object>]	Object to query for legal properties values

Categories

Object, PropertyAndParameter

Description

Gets a list of valid values for an enumerated type property of either a class of objects or a specific object.

Note: The command cannot be used to return valid values for properties other than Enum properties. The `report_property` command will return the type of property to help you identify Enum properties.

Arguments

`-default` - (Optional) Return the default property value for the specified class of objects.

`-class <arg>` - (Optional) Return the property values of the specified class instead of a specific object. The class argument is case sensitive, and most class names are lower case.

Note: `-class` cannot be used together with an `<object>`.

Send Feedback

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - (Required) The name of the property to be queried. Only properties with an enumerated value, or a predefined value set, can be queried with this command. All valid values of the specified property will be returned.

`<object>` - (Optional) A single object on which to report properties.

Note: If you specify multiple objects you will get an error.

Examples

The following example returns the list of valid values for the KEEP_HIERARCHY property from cell objects:

```
list_property_value KEEP_HIERARCHY -class cell
```

The following example returns the same result, but uses an actual cell object in place of the general cell class:

```
list_property_value KEEP_HIERARCHY [get_cells cpuEngine]
```

The following example returns the default value for the specified property by using the current design as a representative of the design class:

```
list_property_value -default BITSTREAM.GENERAL.COMPRESS [current_design]
```

See Also

- create_property
- current_design
- get_cells
- get_property
- list_property
- report_property
- reset_property
- set_property

Send Feedback

list_targets

List applicable targets for the specified source

Syntax

```
list_targets [-quiet] [-verbose] <files>
```

Returns

List of targets.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<files>	Source file for which the targets needs to be listed

Categories

Project

Description

List the targets that are available for a specified IP core, DSP module, or IP Subsystem. The following file types are accepted: .xci, .xco, .mdl, .bd, .bxml.

Use the `generate_targets` command to generate the listed targets.

The command returns the list of available targets. If no targets are available for the specified file objects, nothing is returned.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Send Feedback

`<files>` - (Required) A files object that contains the list of source files to evaluate.

Note: Use `get_files` to specify a files object, rather than specifying a file name.

Examples

The following example lists the available targets for any DSP modules in the design:

```
list_targets [get_files *.mdl]
```

See Also

- create_bd_design
- create_sysgen
- generate_target
- get_files
- import_ip
- read_ip

Send Feedback

load_features

Load Tcl commands for a specified feature.

Syntax

```
load_features [-quiet] [-verbose] [<features>...]
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<features>]	Feature(s) to load, use list_features for a list of available features.

Categories

Tools

Description

Load the specified features of the Vivado Design Suite into memory.

In order to reduce the memory footprint of the Vivado Design Suite, there are groups of Tcl commands called "features" which are unavailable for use until you run a command from that feature set, or unless you explicitly load the feature using the `load_features` command.

For example, the `load_features simulator` command loads the commands for the Vivado simulator, as does directly launching the Vivado simulator using the `launch_xsim` command.

To access the complete list of Tcl commands associated with a feature of the Vivado Design Suite, and the help text for these commands, you can load the feature into the application memory using the `load_features` command without actually running the feature of the tool.

You can list the features that are available to be loaded using the `list_features` command. The list of features is dynamic, and changes from release to release.

The command returns nothing if successful, or an error message if failed.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<features> - List of features to load.

Examples

The following example loads the Vivado simulator feature:

```
load_features simulator
```

The following example loads all of the loadable feature sets of the Vivado Design Suite:

```
load_features [list_features]
```

See Also

- help
- list_features

lock_design

Locks or unlocks netlist, placement or routing of a design. The 'lock/unlock' will only applied on physically placed cells and routed nets

Syntax

```
lock_design [-level <arg>] [-unlock] [-export] [-quiet] [-verbose] [<cell>]
```

Usage

Name	Description
[-level]	specify the locking and unlocking level; Valid values are logical, placement, and routing. Default: placement
[-unlock]	Unlock cells, if cells are not specified, whole design is unlocked; '-level' parameter must be specified for unlocking.
[-export]	mark that the constraints can be exported.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<cell>]	Lock cells, if cells are not specified, whole design is locked. Notice only placed cells and routed nets will be locked. Default: *

Categories

Project

Description

This command is used in the Hierarchical Design Flows for Design Preservation and Partial Reconfiguration. Refer to the *Vivado Design Suite User Guide: Hierarchical Design* (UG905) for more information on these design flows, and the use of this command.

The `lock_design` command is used to lock down the placement and/or routing of a design, or of the specified cell of a design. After reading in an Out-of-Context (OOC) design checkpoint using the `read_checkpoint` command, the preservation level for the module must be defined.

This command sets the IS_LOC_FIXED, IS_BEL_FIXED, and IS_ROUTE_FIXED properties of the specified logic.

Arguments

`-level <arg>` - (Optional) Specify the level of the cell or design to preserve in the current design. As a default, the placement data is preserved. Accepted values are:

Send Feedback

- `logical` - Preserves the logical design. Any placement or routing information is still used, but can be changed if the tools can achieve better results.

- `placement` - Preserves the logical and placed design. Any routing information is still used, but can be changed if the tools can achieve better results. This is the default setting.

- `routing` - Preserves the logical, placed and routed design. Internal routes are preserved, but interface nets are not. In order to preserve routing, the CONTAIN_ROUTING property must have been used on the Pblock during the OOC implementation. This ensures that there will be no routing conflicts when the OOC implementation is reused.

`-unlock` - (Optional) Unlock cells. If cells are not specified, the whole design is unlocked. The `-level` option must be specified for unlocking, just as it was for locking.

> **TIP:** *Locking* `-level routing` *locks the logical, placement, and routing data of the design. However* `-unlock -level routing` *only unlocks the routing data. You must use* `-unlock -level logical` *to unlock the routing, placement, and logical data of the design.*

`-export` - (Optional) Permit the export the placement and routing data as an XDC file. The constraints of a locked design or cell can be exported using the `write_xdc` command. By default, the constraints of a locked design or cell cannot be exported.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<cell>` - (Optional) Lock the specified cells in the design, if cells are not specified, the whole design is locked. The default is to lock all cells in the design.

> **TIP:** *Only placed cells and routed nets will be locked at the level specified.*

Examples

The following example locks the netlist, placement, and routing data for the specified cells of the current design:

```
lock_design -level routing [get_cells usbEngine*]
```

This example unlocks the routing data for the specified cells of the current design, while leaving the netlist and placement data locked from the prior example:

```
lock_design -unlock -level routing [get_cells usbEngine*]
```

The following example unlocks the routing, placement, and netlist data for the specified cells of the current design:

```
lock_design -unlock -level logical [get_cells usbEngine*]
```

See Also

- read_checkpoint
- write_xdc

log_saif

Log Switching Activity Interchange Format (SAIF) toggle for specified wire, signal, or reg

Syntax

```
log_saif [-quiet] [-verbose] <hdl_objects>...
```

Returns

Does not return any object.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hdl_objects>	The hdl_objects to log

Categories

Simulation

Description

Writes the switching activity rates for the specified HDL signals during the current simulation.

The Switching Activity Interchange format (SAIF) file is an ASCII file containing header information, and toggle counts for the specified signals of the design. It also contains the timing attributes which specify time durations for signals at level 0, 1, X, or Z.

The `log_saif` command can only be used after the `open_saif` command has opened an SAIF file in the current simulation to capture switching activity rates.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hdl_objects> - Specifies the HDL signal names on which to capture code.

Examples

The following example logs switching activity for all signals in the current_scope:

```
log_saif [ get_objects ]
```

Log SAIF for only the internal signals starting with name c of the scope `/tb/UUT`:

```
log_saif [get_objects -filter { type == internal_signal }/tb/UUT/c*]
```

See Also

- close_saif
- get_objects
- open_saif

log_vcd

Log Value Change Dump (VCD) simulation output for specified wire, signal, or reg

Syntax

```
log_vcd [-level <arg>] [-quiet] [-verbose] [<hdl_objects>...]
```

Returns

Does not return any object.

Usage

Name	Description
[-level]	Number of levels to log (for HDL scopes) Default: 0
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<hdl_objects>]	Which HDL objects to log

Categories

Simulation

Description

Indicates which HDL objects to write into the Value Change Dump (VCD) file. In some designs the simulation results can become quite large; the `log_vcd` command lets you define the specific content of interest. This command models the behavior of the Verilog `$dumpvars` system task.

HDL objects include HDL signals, variables, or constants as defined in the Verilog or VHDL test bench and source files. An HDL signal includes Verilog wire or reg entities, and VHDL signals. Examples of HDL variables include Verilog real, realtime, time, and event.

This command specifies which HDL objects and how many levels of design hierarchy to write into the VCD file. The actual values of the objects are written to the VCD file when you run the `checkpoint_vcd` or `flush_vcd` commands at a specific time during simulation.

> ⭐ **IMPORTANT!** *You must use the* `open_vcd` *command before using any other* `*_vcd` *commands.*

Nothing is returned by this command.

Arguments

`-level <arg>` - (Optional) Specifies the number of levels of design hierarchy to traverse when locating HDL objects to write to the VCD file. The default value of 0 causes the tool to dump all values for the specified HDL objects at the level of hierarchy defined by <hdl_objects>, and all levels below that. A value of 1 indicates that only the level of hierarchy specified by <hdl_objects> should be written to the VCD file.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hdl_objects> - (Optional) Specifies the HDL objects to identify and write changing values into the VCD file. The level of hierarchy is also represented in the hdl_objects pattern. For instance `/tb/UUT/*` indicates all HDL objects within the `/tb/UUT` level of the design.

Examples

Log value changes for all the ports from the scope `/tb/UUT`:

```
log_vcd [get_objects -filter { type == port } /tb/UUT/* ]
```

Note: Since `-levels` is not specified, all levels below the specified scope will be searched for ports matching the specified pattern as well.

Log VCD for all the objects in the current_scope:

```
log_vcd *
log_vcd [ get_objects *]
```

Log value changes for only internal signals with names starting with C, of the root scope `/tb/UUT`:

```
log_vcd [get_objects -filter { type == internal_signal }./C*]
```

See Also

- checkpoint_vcd
- flush_vcd
- open_vcd

Send Feedback

log_wave

Log simulation output for specified wire, signal, or reg for viewing using Vivado Simulators waveform viewer. Unlike add_wave, this command does not add the waveform object to waveform viewer (i.e. Waveform Configuration). It simply enables logging of output to the Vivado Simulators Waveform Database (WDB).

Syntax

```
log_wave [-recursive] [-r] [-verbose] [-v] [-quiet] <hdl_objects>...
```

Usage

Name	Description
[-recursive]	Searches recursively for objects
[-r]	Searches recursively for objects
[-verbose]	Displays all warnings
[-v]	Displays all warnings
[-quiet]	Ignore command errors
<hdl_objects>	Which hdl_objects to trace

Categories

Simulation

Description

Log simulation activity for the specified HDL objects into the waveform database file (`.wdb`) for viewing using Vivado simulator waveform viewer.

In the Vivado simulator, an HDL object is an entity that can hold a value, such as a wire, signal, or register.

Unlike `add_wave`, this command does not add the waveform object to waveform configuration. It simply enables logging of waveform activity to the Vivado simulator waveform database (WDB). See the *Vivado Design Suite User Guide: Logic Simulation (UG900)* for more information.

This command returns nothing.

Arguments

`-recursive` | `-r` - (Optional) Recursively log the waveform activity of the specified HDL objects, and the children of the specified HDL objects.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hdl_objects> - (Required) Specifies the HDL objects to include in the Vivado simulator waveform database file. The level of hierarchy is also represented in the hdl_objects pattern. For instance `/tb/UUT/*` indicates all HDL objects within the `/tb/UUT` level of the design.

Examples

The following example logs the waveform activities for the specified HDL objects.

```
log_wave -r [get_objects /testbench/dut/*]
```

See Also

- get_objects

ltrace

Turns on or off printing of file name and line number of the hdl statement being simulated

Syntax

```
ltrace [-quiet] [-verbose] <value>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<value>	value: on, true, yes. Otherwise set to off, false, no

Categories

Simulation

Description

Enables line-level tracing for simulation debugging purposes.

During simulation the simulation source file and line number being evaluated is returned to the Tcl console.

> **TIP:** *Process tracing with the `ptrace` command provides more detailed information than is available with line tracing.*

This feature can also be enabled using the LINE_TRACING property on the current simulation object:

```
set_property LINE_TRACING on [current_sim]
```

The command returns the state of line tracing, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

Send Feedback

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<value>` - (Required) Enables or disables line tracing during simulation. Specify a `<value>` of true to enable process tracing, or false to disable it.

Example

The following example enables line tracing:

```
ltrace true
```

See Also

- current_sim
- ptrace
- set_property

make_bd_intf_pins_external

Create external port for the corresponding interface pins. If a cell is specified, create external interface ports for all unconnected interface pins.

Syntax

```
make_bd_intf_pins_external [-quiet] [-verbose] <objects>...
```

Returns

Pass if successful in creating at least one interface port.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<objects>	The interface pins/cells to be made external

Categories

IPIntegrator

Description

Create an external interface port in the current block design and connect that to the selected block interface pin. If a bd_cell is specified as the argument, all unconnected block interface pins of that cell will be made external. The created external interface port will have the same properties as the selected block interface pin.

⭐ **IMPORTANT!** *For a group of block interface pins, one external port will be created per block interface pin.*

The selected block interface pin, if unconnected, will be connected to a new interface port at the top-most level of the block design.

This command returns TCL_OK if it is successful, or TCL_ERROR if it fails, unless `-quiet` is specified.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<objects>` - (Required) Specify either a list of block interface pin objects returned by `get_bd_intf_pins`, or block design cell objects returned by `get_bd_cells`.

Examples

The following example will select the specified block interface pin on the bd_cell `axi_gpio_0`, and connect it to an external interface port:

```
make_bd_intf_pins_external [get_bd_intf_pins axi_gpio_0/S_AXI]
```

The following example will look at all unconnected block interface pins of the bd_cell `axi_gpio_0` and individually connect them to external interface ports:

```
make_bd_intf_pins_external [get_bd_cells axi_gpio_0]
```

See Also

- create_bd_cell
- create_bd_intf_port
- connect_bd_intf_net

Send Feedback

make_bd_pins_external

Create external port for the corresponding pin. If a cell is specified, create external ports for all unconnected pins.

Syntax

```
make_bd_pins_external [-quiet] [-verbose] <objects>...
```

Returns

Pass if successful in creating at least one port.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<objects>	The pins/cells to be made external

Categories

IPIntegrator

Description

Create an external port in the current block design and connect that to the selected block pin. If a bd_cell is specified as the argument, all unconnected block pins of that cell will be made external. The created external port will have the same properties as the selected block pin.

> ⭐ **IMPORTANT!** *For a group of block pins, one external port will be created per block pin.*

The selected block pin, if unconnected, will be connected to a new port at the topmost level of the block design.

This command returns TCL_OK if it is successful, or TCL_ERROR if it fails, unless `-quiet` is specified.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<objects>` - (Required) Specify either a list of block pin objects returned by `get_bd_pins`, or block design cell objects returned by `get_bd_cells`.

Examples

The following example will select a block pin on the bd_cell `axi_gpio_0` and connect it to an external port:

```
make_bd_pins_external [get_bd_pins axi_gpio_0/s_axi_aclk]
```

The following example will look at all unconnected block pins of the bd_cell `axi_gpio_0` and individually connect them to external ports:

```
make_bd_pins_external [get_bd_cells axi_gpio_0]
```

See Also

- create_bd_cell
- create_bd_port
- connect_bd_net

make_diff_pair_ports

Make differential pair for 2 ports

Syntax

```
make_diff_pair_ports [-quiet] [-verbose] <ports>...
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<ports>	Ports to join

Categories

[XDC](#), [PinPlanning](#)

Description

Joins two existing ports to create a differential pair. The port directions, interfaces, and other properties must match in order for the specified ports to be joined as a differential pair. Otherwise an error will be returned.

> ⭐ **IMPORTANT!** *The two ports must first be created, either by using the* `create_port` *command or by reading in an XDC file, prior to making them into a differential pair.*

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<ports> - (Required) Two port objects to join as a differential pair. The first port specified will be the positive side of the differential pair.

Examples

The following example joins the two specified ports to create a differential pair:

```
make_diff_pair_ports port_Pos1 port_Neg1
```

See Also

- create_interface
- create_port
- split_diff_pair_ports

make_wrapper

Generate HDL wrapper for the specified source

Syntax

```
make_wrapper [-top] [-testbench] [-inst_template] [-fileset <arg>]
    [-import] [-force] [-quiet] [-verbose] <files>
```

Usage

Name	Description
[-top]	Create a top-level wrapper for the specified source
[-testbench]	Create a testbench for the specified source
[-inst_template]	Create an instantiation template for the specified source. The template will not be added to the project and will be generated for reference purposes only.
[-fileset]	Fileset name
[-import]	Import generated wrapper to the project
[-force]	Overwrite existing source(s)
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<files>	Source file for which the wrapper needs to be generated

Categories

[Project](), [SysGen]()

Description

Create a Verilog or VHDL wrapper for instantiating a sub-design into the project.

The `make_wrapper` command will create a wrapper for Embedded Processor Designs from the IP integrator of the Vivado Design Suite, or any IP integrator block design, as well as DSP modules created in System Generator or MathWorks MatLab.

You can generate a wrapper to make the sub-design the top-level of a stand-alone design, or for instantiating a sub-design into an existing design. You can also generate a wrapper for a simulation test bench of System Generator sub-designs.

Note: The wrapper is generated in Verilog or VHDL according to the TARGET_LANGUAGE property on the project.

The command returns information related to the creation of the wrappers, or returns an error if it fails.

Arguments

`-top` - (Optional) Create a top-level Verilog or VHDL wrapper for the specified source. The wrapper instantiates the sub-design as the top-level of the design hierarchy.

`-testbench` - (Optional) Create a simulation test bench template for the specified sub-design. This includes the DUT module instantiation, but does not include the stimulus for simulation.

> ⭐ **IMPORTANT!** *This option is only valid for SysGen composite files. All other sources will produce an error.*

`-inst_template` - (Optional) Create an instantiation template for the specified source. The template will not be added to the project and will be generated for reference purposes only. The instantiation template can be cut and paste into another RTL file to create an instance of the module in the hierarchy.

`-fileset` - (Optional) Specify the destination fileset for importing the wrapper file into the project. By default, the wrapper will be imported into sources_1.

`-import` - (Optional) Import the wrapper file into the project, adding it to the appropriate fileset.

`-force` - (Optional) Overwrite an existing wrapper file.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<files>` - (Required) Specify the source files to generate the wrapper from. The `make_wrapper` command only supports the `.mdl` file format from System Generator for DSP, the `.slx` format from MathWorks MATLAB, and the `.bd` file format from the IP integrator of the Vivado Design Suite.

Examples

The following example creates the instantiation template to integrate the specified IP integrator block design into the design hierarchy of the current project:

```
make_wrapper -inst_template -fileset [get_filesets sources_1] \
-files [get_files C:/Data/design_1/design_1.bd]
```

See Also

- add_files
- create_bd_design
- create_sysgen
- generate_target
- get_filesets
- list_targets

mark_objects

Mark objects in GUI

Syntax

```
mark_objects [-rgb <args>] [-color <arg>] [-quiet] [-verbose] <objects>
```

Usage

Name	Description
`[-rgb]`	RGB color index list
`[-color]`	Valid values are red green blue magenta yellow cyan and orange
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<objects>`	Objects to mark

Categories

GUIControl

Description

Marks specified objects in GUI mode. This command places an iconic mark to aid in the location of the specified object or objects. The mark is displayed in a color as determined by one of the color options.

Objects can be unmarked with the `unmark_objects` command.

Note: Use only one color option. If both color options are specified, `-rgb` takes precedence over `-color`.

Arguments

`-rgb` <args> - (Optional) The color to use in the form of an RGB code specified as {R G B}. For instance, {255 255 0} specifies the color yellow, while {0 255 0} specifies green.

`-color <arg>` - (Optional) The color to use for marking the specified object or objects. Supported colors are: red, green, blue, magenta, yellow, cyan, and orange.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

Send Feedback

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<objects>` - (Required) One or more objects to be marked.

Examples

The following example adds a red icon to mark the currently selected objects:

```
mark_objects -color red [get_selected_objects]
```

See Also

- get_highlighted_objects
- get_marked_objects
- get_selected_objects
- highlight_objects
- select_objects
- unmark_objects

modify_debug_ports

Modify routed probe connections to debug cores.

Syntax

```
modify_debug_ports [-probes <args>] [-quiet] [-verbose]
```

Usage

Name	Description
[-probes]	List of probes to be connected: debug core pin, channel index, and logical net for each probe connection.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Debug

Description

Modifies a routed design to connect nets to specified ports of debug cores. This command takes a list of connections to be made to specified debug probes. Each connection is defined as a Tcl list, enclosed in braces {}, specifying the following three elements separated by spaces:

1. The logical pin of the debug core to be connected.

2. The channel index of the specified probe.

3. The logical net of the signal to be probed.

Multiple probe connections are specified as a list of lists, with each connection itself being a Tcl list as shown in the example.

The command performs all of the netlist modifications to disconnect existing net connections to the specified probe ports as needed, connecting each net to be probed to the specified probe port, and automatically routing the modified connections. Nets that become disconnected during the process are left unconnected.

Arguments

`-probes` <args> - (Required) Specifies a list of probe connections as a Tcl list of lists. Each probe connection is defined as a triplet of three elements in the following order separated by spaces: 1) the logical pin of the debug core to be connected, 2) the probe channel index, and 3) the logical net of the signal to be probed. Multiple probe connections are specified as a list of lists as follows:

```
[list  \{probe1 channel1 net1}  \{probe2 channel1 net2} \{probe2 channel2
net3} ]
```

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example modifies 3 probe connections:

```
modify_debug_ports -probes [list {top/x_ila/probe0 0 top/inst_A/net_0} \
   {top/x_ila/probe1 1 top/inst_A/net_a} {top/x_ila/probe1 2 top/inst_A/
net_b}]
```

> **TIP:** *The* `modify_debug_ports` *command moves a port probe from one signal to another.*

See Also

- connect_debug_port
- create_debug_core
- create_debug_port
- disconnect_debug_port
- set_property

move_bd_cells

Move cells into a hierarchy cell. The connections between these cells are maintained; the connections between these cells and other cells are maintained through crossing hierarchy cell.

Syntax

```
move_bd_cells [-prefix <arg>] [-quiet] [-verbose] [<parent_cell>]
    [<cells>...]
```

Returns

0 if success.

Usage

Name	Description
[-prefix]	Prefix name to add to cells
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<parent_cell>]	Parent cell
[<cells>]	Match engine names against cell names Default: *

Categories

IPIntegrator

Description

Move IP integrator cells into the specified hierarchical module within the current subsystem design. The connections between the cells being moved are maintained; connections between these cells and other cells that are not being moved are maintained automatically by IP integrator adding pins and ports to cross the hierarchical boundary.

Cells in the IP subsystem design can also be copied into a hierarchical module using `copy_bd_objs`, and can be grouped and added to a hierarchical module using `group_bd_cells`.

This command returns the name of the `<parent_cell>` module when successful, or returns an error message if it failed.

Arguments

`-prefix <arg>` - (Optional) A prefix name to apply to any cells that are moved into the hierarchical module.

Send Feedback

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<parent_cell>` - The name of the hierarchical module to move cells into.

`<cells>` - (Optional) The list of cells, specified by the `get_bd_cells` command, to move from the current IP subsystem design into the hierarchical module.

Example

The following example:

```
move_bd_cells -prefix mod1_ /myModule1 [get_bd_cells /myAxiFifo_1]
/myModule1
```

See Also

- copy_bd_objs
- create_bd_cell
- group_bd_cells

move_dashboard_gadget

Reposition the Gadget for Project summary dashboard

Syntax

```
move_dashboard_gadget -name <arg> -row <arg> -col <arg> [-dashboard <arg>]
    [-quiet] [-verbose]
```

Usage

Name	Description
-name	Name of the gadget
-row	Target row number to which the gadget has to be moved
-col	Target column number to which the gadget has to be moved
[-dashboard]	Name of the dashboard to which the gadget is associated
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Project

Description

Specify the placement of a dashboard gadget into the dashboard.

> 💡 **TIP:** *Currently the Project Summary is the only dashboard, so the gadget is placed into that dashboard.*

The arrangement of gadgets in a dashboard is by row and column. The Project Summary dashboard has two columns, 0 and 1, and has as many rows as needed to display all the gadgets. If you specify a column other than 0 or 1, or specify a row in excess of the current defined rows + 1, an error will be returned.

Arguments

-name <arg> - (Required) Specify the name of the gadget to move.

-row <arg> - (Required) Specify the row in the Project Summary dashboard for the gadget placement.

-col <arg> - (Required) Specify the column in the Project Summary dashboard for the gadget placement.

Send Feedback

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Example

The following example moves the specified gadget to column 0, row 3:

```
move_dashboard_gadget -name utilization_1 -row 3 -col 0
```

See Also

- create_dashboard_gadget
- delete_dashboard_gadgets
- get_dashboard_gadgets

Send Feedback

move_files

Moves the files from one fileset to another while maintaining all of their original properties.

Syntax

```
move_files [-fileset <arg>] [-of_objects <args>] [-quiet] [-verbose]
    [<files>...]
```

Returns

List of files that were moved.

Usage

Name	Description
[-fileset]	Destination fileset name
[-of_objects]	Reconfig Modules to move the files to
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<files>]	Name of the files to be moved

Categories

Project, Simulation

Description

Moves files returned by the `get_files` command from one fileset to another while maintaining the properties on the files.

This command returns the list of files that were moved, or an error if the command fails.

Arguments

`-fileset <arg>` - (Optional) The destination fileset to which the specified source files should be moved. If no fileset is specified the files are moved to the sources_1 fileset by default. An error is returned if the specified fileset does not exist.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

Send Feedback

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<files>` - (Optional) One or more file objects returned by the `get_files` command.

> ⭐ **IMPORTANT!** *You must specify file objects returned by the `get_files` command, and not simply specify file names.*

Examples

The following example moves the file, `top_full.xdc`, to the constrs_2 fileset.

```
move_files -fileset constrs_2 [get_files top_full.xdc]
```

See Also

- get_files

move_wave

Moves wave objects from their current position to the specified position in the wave configuration

Syntax

```
move_wave [-into <args>] [-at_wave <args>] [-after_wave <args>]
    [-before_wave <args>] [-quiet] [-verbose] <items>...
```

Usage

Name	Description
[-into]	the wave configuration, group, or virtual bus into which the wave object(s) will be moved.
[-at_wave]	inserts the new wave object(s) into the specified wave object, or after the specified wave object if not a group or virtual bus
[-after_wave]	inserts the new wave objects(s) after the specified wave object
[-before_wave]	inserts the new wave objects(s) before the specified wave object
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<items>	wave objects to move

Categories

Waveform

open_bd_design

Open an existing IP subsystem design from disk file.

Syntax

```
open_bd_design [-quiet] [-verbose] <name>
```

Returns

The design object. Returns nothing if the command fails.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<name>	Name of IP subsystem design to open

Categories

IPIntegrator

Description

Open an IP subsystem design in the IP integrator of the Vivado IDE. The IP subsystem must previously have been created using the `create_bd_design` command.

This command returns a message with the name of the opened IP subsystem design, or returns an error if the command fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - The path and file name of the IP subsystem design to open in the IP integrator of the Vivado Design Suite. The name must include the file extension.

Examples

The following opens the specified IP subsystem design in the current project:

```
open_bd_design  C:/Data/project1/project1.src/sources_1/bd/design_1/
design_1.bd
```

See Also

- close_bd_design
- create_bd_design
- current_bd_design
- save_bd_design

open_checkpoint

Open a design checkpoint in a new project

Syntax

```
open_checkpoint [-part <arg>] [-ignore_timing] [-quiet] [-verbose] <file>
```

Usage

Name	Description
[-part]	Override the checkpoint part. Note that this may cause errors if the checkpoint contains xdef.
[-ignore_timing]	reload a checkpoint without the timing constraints. Will not work if '-time' option given.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<file>	Design checkpoint file

Categories

Project

Description

Open a design checkpoint file (DCP), create a new in-memory project and initialize a design immediately in the new project with the contents of the checkpoint. This command can be used to open a top-level design checkpoint, or the checkpoint created for an out-of-context module.

When opening a checkpoint, there is no need to create a project first. The `open_checkpoint` command reads the design data into memory, opening the design in Non-Project Mode. Refer to the *Vivado Design Suite User Guide: Design Flows Overview* (UG892) for more information on Project Mode and Non-Project Mode.

Note: When multiple design checkpoints are open in the Vivado tool, you must use the `current_project` command to switch between the open designs. You can use `current_design` to check which checkpoint is the active design.

Arguments

`-part <arg>` - (Optional) Specify a target part for the imported checkpoint design. This option lets you change the speed grade of the part used by the design checkpoint file, or change early availability parts for production parts of the same device and package.

> **IMPORTANT!** *The use of `-part` is limited in terms of the range of parts that can be used, and can result in an error when opening the checkpoint if an incompatible part is specified.*

`-ignore_timing` - (Optional) Open the checkpoint without the timing constraints to make opening the checkpoint quicker.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file>` - (Required) The path and filename of the checkpoint file.

Note: If the path is not specified as part of the file name, the tool will search for the specified file in the current working directory and then in the directory from which the tool was launched.

Examples

The following example opens the specified checkpoint file, and specifies the target part for the design:

```
open_checkpoint C:/Data/state1/checkpoint.dcp -part xc7k325tffg900-2
```

Note: If the specified part is not compatible with the device and package used by the specified checkpoint, the command will return an error.

See Also

- current_design
- current_project
- read_checkpoint
- write_checkpoint

open_example_project

Open the example project for the indicated IP

Syntax

```
open_example_project [-dir <arg>] [-force] [-in_process] [-quiet]
    [-verbose] <objects>...
```

Returns

The Project that was opened.

Usage

Name	Description
[-dir]	Path to directory where example project will be created
[-force]	Overwrite an example project if it exists
[-in_process]	Open the example project in the same process
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<objects>	The objects whose example projects will be opened

Categories

Project, IPFlow

Description

Open an example project for the specified IP cores. The example project can be used to explore the features of the IP core in a stand-alone project, instead of integrated into the current project.

Arguments

-dir <arg> - (Optional) Specifies the path to the directory where the example project will be written.

-force - (Optional) Force the opening of a new example project, overwriting an existing example project at the specified path.

-in_process - (Optional) Open the example project in the same tool process as the current project. As a default, without this argument, a new process instance of the tool will be launched for the example project.

Send Feedback

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<objects>` - (Required) The IP cores to open example projects for.

Examples

The following copies the IP customization and opens the example project for the specified IP core in a new location:

```
open_example_project -dir C:/Data/examples -force [get_ips blk_mem*]
```

See Also

- create_ip
- generate_target
- get_ips
- import_ip

open_hw_manager

Open the hardware tool

Syntax

```
open_hw_manager [-quiet] [-verbose]
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Hardware

Description

Open the Hardware Manager in the Vivado Design Suite in either the Vivado IDE or in Tcl or batch mode. Opening the Hardware Manager is the first step in programming and/or debugging your design in Xilinx FPGA hardware. For more information refer to the *Vivado Design Suite User Guide: Programming and Debugging (UG908)*.

Hardware Manager is a feature of the Vivado Design Suite which lets you interact with FPGA devices on a board. The features of the Hardware Manager include:

- Logic Debug or Logic Analyzer- Debugging FPGA programmable logic designs.

- Programming/Configuration - Program FPGA devices using JTAG and configuring flash memory devices connected to FPGAs.

- In-system Serial I/O debug - Adjust SERDES receive/transmit settings and measure transmission bit error rates.

- System Monitor - Control on chip system monitor and read system monitor temperature and voltage values.

The Hardware Manager uses a number of first class objects, like hw_server, hw_target, hw_device, and hw_ila. Each of these objects is related to other objects, and has properties that can be set or read by the `set_property` and `get_property` commands to configure or control its function in the Hardware Manager. Refer to the *Vivado Design Suite Properties Reference Guide (UG912)* for more information on these objects.

The steps to connect to hardware and program the target FPGA are:

Send Feedback

1. Open the hardware manager in the IDE (`open_hw_manager`).

> 💡 **TIP:** *This step can be skipped if you are running in batch or Tcl mode.*

2. Connect to a hardware server running either on the local machine, or on a remote network accessible host (`connect_hw_server`).

3. Open a hardware target on the connected hardware server (`open_hw_target`).

4. Identify the Xilinx FPGA on the open hardware target (`current_hw_device`, `get_hw_devices`).

5. Associate the bitstream data programming file (`.bit`), and probes file (`.ltx`) if one exists, with the appropriate FPGA (`set_property`).

6. Program or download the programming file into the hardware device (`program_hw_device`, `refresh_hw_device`).

Note that you can run the Hardware Manager from within the Vivado tool without having a project or design open. You can open the Hardware Manager, connect to the hardware server, and program the device on the target by providing a bitstream file, and probes file for debugging.

You can close the Hardware Manager using the `close_hw_manager` command.

This command returns nothing if successful, and returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Example

The following example opens the Hardware Manager in the Vivado Design Suite:

```
open_hw_manager
```

See Also

- close_hw_manager
- connect_hw_server

Send Feedback

- current_hw_device
- current_hw_server
- get_hw_devices
- get_hw_targets
- open_hw_target
- refresh_hw_device
- set_property

open_hw_platform

Open the Xilinx Shell Archive

Syntax

```
open_hw_platform [-no_auto_upgrade] [-quiet] [-verbose] [<file>]
```

Returns

The name of the shell file.

Usage

Name	Description
[-no_auto_upgrade]	Disable BD auto upgrade
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<file>]	Xilinx Shell Archive file Values: A filename with alphanumeric characters and .dsa/.xsa extension.

Categories

Vitis, Platform, FileIO

Description

Open a Xilinx support archive (XSA) file and extract the Vivado project, block design, and IP from the archive. This will create a project directory and project file (`.xpr`) from the XSA.

Note: The project will be created in the current working directory, or the directory from which the Vivado tool was launched.

This command returns a transcript of its actions, or returns an error if it fails.

Arguments

`-no_auto_upgrade` - (Optional) Do not upgrade the IP used in the Block Design to the latest version. By default, the original IP from the XSA will be auto-upgraded if possible. Use the `report_ip_status` command to identify out-of-date IP.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

Send Feedback

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file>` - (Required) Specify the path and file name of the XSA file.

Note: If the path is not specified as part of the file name, the tool will search for the specified file in the current working directory and then in the directory from which the tool was launched.

Examples

The following example opens the specified XSA, automatically upgrading the IP as needed:

```
open_hw_platform C:/Data/zc706.xsa
```

See Also

- validate_hw_platform
- write_hw_platform

open_hw_target

Open a connection to a hardware target on the hardware server

Syntax

```
open_hw_target [-jtag_mode <arg>] [-xvc_url <arg>] [-auto_calibrate]
    [-quiet] [-verbose] [<hw_target>]
```

Usage

Name	Description
[-jtag_mode]	Open target in JTAG mode
[-xvc_url]	Open target connection to XVC server
[-auto_calibrate]	Auto-calibrate target for optimal frequency (SmartLynq cable only)
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<hw_target>]	hardware target Default: current hardware target

Categories

[Hardware](#)

Description

Opens a connection to the specified hardware target of the connected hardware servers, or opens the current hardware target.

The hardware target is a system board containing a JTAG chain of one or more Xilinx devices that you can program with a bitstream file, or use to debug your design. Connections between hardware targets on the system board and the Vivado Design Suite are managed by the Xilinx hardware server application, and the `connect_hw_server` command. Refer to *Vivado Design Suite User Guide: Programming and Debugging (UG908)* for a list of supported JTAG download cables and devices.

Each hardware target can have one or more Xilinx devices to program, or to use for debugging purposes. The current device is specified or returned by the `current_hw_device` command.

Use the `open_hw_target` command to open a connection to one of the available hardware targets. The open target is automatically defined as the current hardware target. Alternatively, you can define the current target with the `current_hw_target` command, and then open the current target. The Vivado Design Suite directs programming and debug commands to the open target through the hardware server connection.

An open connection to the hardware target can be closed using the `close_hw_target` command.

The `open_hw_target` command returns connection messages from the hardware server, or returns an error if it fails.

Arguments

`-jtag_mode [ on | off ]` - (Optional) Open the hardware target in JTAG mode for accessing boundary scan. When the target is running in JTAG mode the Instruction Register (IR) and Data Registers (DR) are accessible through the `scan_ir_hw_jtag` and `scan_dr_hw_jtag` commands, and the devices on the target can also be put into various states using the `run_state_hw_jtag` command.

Note: This is a boolean property of the `hw_target` object that can also be enabled using values of "1" or "True".

`-xvc_url arg` - (Optional) Open the hardware target as a connection to a Xilinx Virtual Cable. Xilinx Virtual Cable (XVC) is a TCP/IP-based protocol that acts like a JTAG cable, and lets you access and debug your Xilinx device without using a physical cable. The argument takes the IP address and port number of the xvcServer application.

Note: Refer to https://www.xilinx.com/products/intellectual-property/xvc.html for more information.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_target> - (Optional) Specify the hw_target object to open a connection to for programming and debug. The hw_target must be specified as an object as returned by the `get_hw_targets` or `current_hw_target` commands. If no target is specified, the Vivado Design Suite will open a connection to the `current_hw_target`.

Example

The following example opens the hardware target returned by the `get_hw_targets` command:

```
open_hw_target [lindex [get_hw_targets] 0]
```

Send Feedback

The following example shows the flow of opening the Hardware Manager, making a connection to the hw_server application running on a remote host, setting the current hardware target, and opening that target, with JTAG mode enabled:

```
open_hw
connect_hw_server -url jupiter:3121
current_hw_target [get_hw_targets */xilinx_tcf/Digilent/210203327463A]
open_hw_target -jtag_mode on
```

See Also

- close_hw_target
- connect_hw_server
- current_hw_device
- current_hw_target
- get_hw_devices
- get_hw_targets

Send Feedback

open_io_design

Open an IO design

Syntax

```
open_io_design [-name <arg>] [-part <arg>] [-constrset <arg>] [-quiet]
    [-verbose]
```

Returns

Design object

Usage

Name	Description
[-name]	Design name
[-part]	Target part
[-constrset]	Constraint fileset to use
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Project](#)

Description

Opens a new or existing I/O Pin Planning design.

Note: The design_mode property for the current source fileset must be defined as PinPlanning in order to open an I/O design. If not, you will get the following error:

```
ERROR: The design mode of 'sources_1' must be PinPlanning
```

Arguments

`-name <arg>` - (Optional) The name of a new or existing I/O Pin Planning design.

`-part <arg>` - (Optional) The Xilinx device to use when creating a new design. If the part is not specified the default part will be used.

`-constrset <arg>` - (Optional) The name of the constraint fileset to use when opening an I/O design.

Note: The `-constrset` argument must refer to a constraint fileset that exists. It cannot be used to create a new fileset. Use `create_fileset` for that purpose.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following creates a new I/O design called myIO:

```
open_io_design -name myIO
```

Note: The default source set, constraint set, and part will be used in this case.

The following example opens an existing I/O design called myIO, and specifies the constraint set to be used:

```
open_io_design -name myIO -constrset topCon
```

See Also

- create_project

open_project

Open a Vivado project file (.xpr)

Syntax

```
open_project [-part <arg>] [-read_only] [-quiet] [-verbose] <file>
```

Returns

Opened project object.

Usage

Name	Description
[-part]	Open the project using this part (overrides project's part)
[-read_only]	Open the project in read-only mode
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<file>	Project file to be read

Categories

Project

Description

Opens the specified Vivado Design Suite project file (`.xpr`), or the project file for the Vivado Lab Edition (`.lpr`).

> ⭐ **IMPORTANT!** *The `open_project` command has a different command syntax in the Vivado Lab Edition. The `-part` option is not supported because the Vivado Lab Edition project (`.lpr`) does not specify a target part. The `current_hw_target` and `current_hw_device` commands determine the target part.*

This command returns a transcript of its process and the name of the created project, or returns an error if it fails.

Arguments

`-part <arg>` - (Optional) Specify a target part to use when opening the project. This option lets you change the speed grade of the part used by a saved project, or change early availability parts for production parts of the same device and package. This option is not supported in Vivado Lab Edition.

Send Feedback

> ⭐ **IMPORTANT!** *The use of `-part` is limited in terms of the range of parts that can be used, and can result in an error when opening the project if an incompatible part is specified.*

`-read_only` - (Optional) Open the project in read only mode. You will not be able to save any modifications to the project unless you use the `save_project_as` command to save the project to a new editable project.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file>` - (Required) The project file to open. You must include both the path to the file and the `.xpr` file extension.

Examples

The following example opens the project named `my_project1` located in the Designs directory.

```
open_project C:/Designs/project1.xpr
```

Note: The project must be specified with the `.xpr` extension for the tool to recognize it as a project file. The path to the file must be specified along with the project file name or the tool will return an error that it cannot find the specified file.

See Also

- close_project
- create_project
- current_hw_device
- current_hw_target
- current_project

open_report

Open report from .rpx file

Syntax

```
open_report [-file <arg>] [-append] [-console] [-name <arg>]
    [-return_string] [-quiet] [-verbose] <rpx>
```

Usage

Name	Description
[-file]	Filename to output results to
[-append]	Append the results to file, don't overwrite the results file
[-console]	Send output to console
[-name]	Output the results to GUI panel with this name
[-return_string]	Return report as string
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<rpx>	Report data file to be read

Categories

[Report](#)

Description

Read an RPX (protobuf) file into memory to reload report results into the Vivado Design Suite. This command requires an open implemented or synthesized design.

The RPX file is written by report commands such as `report_timing_summary`, and `report_pulse_width`, that support the `-rpx` option, and is an interactive report file that can be reloaded into memory. Reloading the report into memory, reconnects the objects in the report to design objects so that cross-selection between the report in the Vivado IDE and the design is enabled.

This command returns the report results to the Tcl console by default, or when `-console` is specified, or opens a report window in the Vivado IDE when `-name` is specified. This command returns an error if it fails.

Arguments

`-file <arg>` - (Optional) Write the report into the specified file. The specified file will be overwritten if one already exists, unless `-append` is also specified.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-console` - (Optional) Output the report results to the Tcl console in the Vivado IDE or Tcl shell mode. This is the default behavior of the `open_report` command if no other options are specified.

`-name <arg>` - (Optional) Specifies the name of the results set for the GUI. Timing summary reports in the GUI can be deleted by the `delete_timing_results` command.

> 💡 **TIP:** *Opening the RPX file in a named window in the VIvado IDE links objects between the report and the design for cross-selection.*

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<rpx> - Specify the name of the RPX file to load into memory.

Note: If the path is not specified as part of the file name, the tool will search for the specified file in the current working directory and then in the directory from which the tool was launched.

Examples

The following example reads the specified RPX file an opens a named report in the Vivado IDE:

```
open_report -name RPX1 design1_summary.rpx
```

See Also

- check_timing
- report_bus_skew
- report_config_timing

- report_datasheet
- report_drc
- report_methodology
- report_power
- report_pulse_width
- report_timing
- report_timing_summary

Send Feedback

open_run

Open a run into a netlist or implementation design

Syntax

```
open_run [-name <arg>] [-pr_config <arg>] [-quiet] [-verbose] <run>
```

Returns

Design object

Usage

Name	Description
[-name]	Design name
[-pr_config]	PR Configuration to apply while opening the design (only valid when opening a synthesis run)
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<run>	Run to open into the design

Categories

Project

Description

Opens the specified synthesis run into a Netlist Design or implementation run into an Implemented Design. The run properties defining the target part and constraint set are combined with the synthesis or implementation results to create the design view in the tool.

This command is intended to open a synthesized or implemented design that was created from design runs in Project Mode in the Vivado Design Suite.

Use the `open_checkpoint` command to open a Non-Project based checkpoint into memory, opening the design in Non-Project Mode. Refer to the *Vivado Design Suite User Guide: Design Flows Overview* (UG892) for more information on Project Mode and Non-Project Mode.

Arguments

`-name` - (Optional) This is the name assigned to the synthesized or implemented design when the run is opened by the Vivado tool. This name is for reference purposes, and has nothing to do with the top-level of the design or any logic contained within.

Send Feedback

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<run>` - (Required) Specifies the run name of the synthesis or implementation run to open. The run must have completed synthesis or implementation before it can be opened as a design.

Note: If you attempt to open a run that has not been launched the tool will return an error.

Examples

The following command opens the specified synthesis run into a Netlist Design named synthPass1:

```
open_run -name synthPass1 synth_1
```

The following opens an Implemented Design for impl_1:

```
open_run impl_1
```

See Also

- launch_runs
- link_design
- open_checkpoint
- read_checkpoint
- write_checkpoint

open_saif

Open file for storing signal switching rate for power estimation. The switching rate is written out in Switching Activity Interchange Format (SAIF) Only one SAIF is allowed to be open per simulation run.

Syntax

```
open_saif [-quiet] [-verbose] <file_name>
```

Returns

The SAIF object that was opened.

Usage

Name	Description
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<file_name>`	The SAIF filename to store information

Categories

Simulation

Description

Create or open a Switching Activity Interchange Format (SAIF) file for storing signal switching rates in the current simulation for later use by the `report_power` command.

The Switching Activity Interchange format (SAIF) file is an ASCII file containing header information, and toggle counts for the specified signals of the design. It also contains the timing attributes which specify time durations for signals at level 0, 1, X, or Z.

The SAIF file is recommended for power analysis since it is smaller than the VCD file.

When an SAIF file has been opened, you can write the switching activity from the simulation into the SAIF file using `log_saif`.

Only one SAIF can be open at one time during simulation. To close the SAIF file, use the `close_saif` command.

This command returns the object ID of the opened SAIF file, or returns an error if the command failed.

Arguments

`-quiet` - Execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

`-verbose` - Suspends message limits during command execution.

`<file_name>` - Specifies the name of the SAIF file to open.

Examples

The following example opens the specified simulation:

```
open_saif myData.saif
```

See Also

- close_saif
- log_saif

open_vcd

Open a Value Change Dump (VCD) file for capturing simulation output. This Tcl command models behavior of $dumpfile Verilog system task

Syntax

```
open_vcd [-quiet] [-verbose] [<file_name>]
```

Returns

Returns a Vcd Object and sets the current vcd to this object so that current_vcd command will return this object.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<file_name>]	file name. Defaults to dump.vcd (This is LRM standard) Default: dump.vcd

Categories

Simulation

Description

Create or open a Value Change Dump (VCD) file to capture simulation output. This command operates like the Verilog `$dumpfile` simulator directive.

VCD is an ASCII file containing header information, variable definitions, and value change details of a set of HDL signals. The VCD file can be used to view simulation result in a VCD viewer or to estimate the power consumption of the design.

When a VCD file has been opened, you can write the value changes from the simulation into the VCD file using `checkpoint_vcd`, `flush_vcd`, or `log_vcd`. In addition, you can pause and resume the collection of value change data with the `stop_vcd` and `start_vcd` commands.

You can limit the size of the VCD file by using the `limit_vcd` command.

To close the VCD file, use the close_vcd command.

Note: You must use the `open_vcd` command before using any other `*_vcd` commands. Only one VCD file can be open at any time.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file_name>` - (Optional) Is the name of the file into which to dump the current VCD information. When a filename is not specified, the default filename of `dump.vcd` is used. If the specified VCD file already exists, then `open_vcd` resets the VCD file to a new state, overwriting the current contents.

Examples

The following example opens the specified VCD file (`design1.vcd`) so that value changes can be written to it. The `log_vcd` command identifies all ports in the `/tb/UUT` scope, and only that level of the design hierarchy, to be written to the VCD file. The simulation is run for a specified period of time, and `flush_vcd` writes the current values of the HDL objects to the VCD file. Then `close_vcd` closes the open file.

```
open_vcd design1.vcd
log_vcd -level 1 [get_objects filter { type == port } /tb/UUT/* ]
run 1000
flush_vcd
close_vcd
```

See Also

- checkpoint_vcd
- close_vcd
- flush_vcd
- limit_vcd
- log_vcd
- start_vcd
- stop_vcd

Send Feedback

open_wave_config

Open a wave config

Syntax

```
open_wave_config [-quiet] [-verbose] [<filename>]
```

Returns

The wave config opened.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<filename>]	the name of a WCFG file from which to create a new wave configuration and corresponding wave window

Categories

[Waveform](#)

Description

Open the specified Wave Config file (`.wcfg`) in the current simulation.

Vivado simulator uses a simulation debug data model to allow users to debug HDL source files using source code stepping, breakpoints, conditions, and waveform viewing tools. The debug data model contains HDL object and scope names, and maps them to memory addresses to let you examine the changing values of signals, variables and constants during the simulation.

The waveform database is separate from the Wave Config file that stores the waveform activity for the simulation. The Wave Config file contains just the list of wave objects (signals, dividers, groups, virtual buses) to display, and their display properties, plus markers. The waveform database (WDB) contains the event data, values changing over time, for all traced signals, whether displayed or not.

A wave configuration object is created in the current simulation with the `create_wave_config` command. A Wave Config file is written to disk by the use of the `save_wave_config` command, and can be opened with the `open_wave_config` command.

The `open_wave_config` command opens a Wave Config file and maps it to the data source in the current simulation.

Send Feedback

> **IMPORTANT!** *Any HDL objects that are specified in the Wave Config file that are not found in the current simulation will be ignored.*

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<filename> - (Optional) Specify the path and <filename>of the Wave Config file to open.

The <filename> should have a suffix of `.wcfg`, and the file suffix will be assigned automatically if no other suffix is supplied.

Note: If the path is not specified as part of the file name, the tool will search for the specified file in the current working directory and then in the directory from which the tool was launched.

Examples

The following example opens the specified Wave Config file:

```
open_wave_config testbench.wcfg
```

See Also

- close_wave_config
- create_wave_config
- open_wave_database
- save_wave_config

Send Feedback

open_wave_database

Open Waveform Database (WDB) file produced by a prior simulation run and return a simulation object

Syntax

```
open_wave_database [-noautoloadwcfg] [-protoinst <args>] [-quiet]
    [-verbose] <wdb>
```

Usage

Name	Description
[-noautoloadwcfg]	Do not automatically open associated WCFG files
[-protoinst]	Specify a .protoinst file for protocol analysis
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<wdb>	file name

Categories

Simulation

Description

The `open_wave_database` command opens an existing static simulator database file (WDB) and associated wave config file (WCFG). This simulation is a static simulation, not live, and can only be used to review prior results.

Note: Many of the commands for running and resetting the simulation are not available in a static simulation.

Vivado simulator uses a simulation debug data model to allow users to debug HDL source files using source code stepping, breakpoints, conditions, and waveform viewing tools. The debug data model contains HDL object and scope names, and maps them to memory addresses to let you examine the changing values of signals, variables and constants during the simulation. When the simulation completes, the simulation is written to a static simulator database file (WDB).

HDL objects can be added to the simulation waveform database using the `log_wave` command which enables logging of waveform activity for the specified objects to the Vivado simulator waveform database.

Send Feedback

The waveform database is associated with a Wave Config file that stores the waveform activity for the simulation. The Wave Config file contains just the list of wave objects (signals, dividers, groups, virtual buses) to display, and their display properties, plus markers. The waveform database (WDB) contains the event data, values changing over time, for all traced signals, whether displayed or not.

A Wave Config file is written to disk by the use of the `save_wave_config` command, and can be opened with the `open_wave_config` command.

Use the `open_wave_database` command with the `open_wave_config` command to open a previously completed simulation for review in the Vivado IDE.

> **TIP:** *Objects that were logged in the simulation waveform database, with the `log_wave` command, can be added posthumously to the wave configuration in a static simulation using the `add_wave` command.*

Arguments

`-noautoloadwcfg` - (Optional) Do not automatically open wave config files (WCFG) associated with the waveform database.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<wbd> - specifies the path and filename of the WDB file to open.

Examples

The following example opens a WDB file with the specified name, then opens an associated Wave Config file, and finally uses the `current_fileset` command to open the simulation database in the Vivado IDE:

```
open_wave_database {C:/Data/project_xsim/testbench_behav.wdb}
open_wave_config {C:/Data/project_xsim/testbench_behav.wcfg}
current_fileset
```

See Also

- create_wave_config
- current_fileset
- get_wave_configs

Send Feedback

- [open_wave_config](#)
- [save_wave_config](#)

opt_design

Optimize the current netlist. This will perform the retarget, propconst, sweep and bram_power_opt optimizations by default.

Syntax

```
opt_design [-retarget] [-propconst] [-sweep] [-bram_power_opt] [-remap]
    [-aggressive_remap] [-resynth_remap] [-resynth_area]
    [-resynth_seq_area] [-directive <arg>] [-muxf_remap]
    [-hier_fanout_limit <arg>] [-bufg_opt] [-mbufg_opt]
    [-shift_register_opt] [-dsp_register_opt] [-srl_remap_modes <arg>]
    [-control_set_merge] [-merge_equivalent_drivers] [-carry_remap]
    [-debug_log] [-property_opt_only] [-quiet] [-verbose]
```

Usage

Name	Description
[-retarget]	Retarget
[-propconst]	Propagate constants across leaf-level instances
[-sweep]	Remove unconnected leaf-level instances
[-bram_power_opt]	Perform Block RAM power optimizations
[-remap]	Remap logic optimally in LUTs
[-aggressive_remap]	High effort remap optimization
[-resynth_remap]	High effort re-synthesis to reduce the depth of logic in critical timing paths
[-resynth_area]	Resynthesis
[-resynth_seq_area]	Resynthesis (with Sequential optimizations)
[-directive]	Mode of behavior (directive) for this command. Please refer to Arguments section of this help for values for this option Default: Default
[-muxf_remap]	Optimize all MuxFx cells to LUT3
[-hier_fanout_limit]	Replicate by module with threshold N
[-bufg_opt]	Insert, Merge and Split BUFGs
[-mbufg_opt]	Merge BUFGs to MBUFGs
[-shift_register_opt]	Pull register stage from shift register
[-dsp_register_opt]	Push/Pull Registers out of a DSP
[-srl_remap_modes]	remap shift registers to flops or flops to shift registers
[-control_set_merge]	Merge all equivalent control set drivers to a single driver
[-merge_equivalent_drivers]	Merge all LUT,Flop equivalent driver replications
[-carry_remap]	reamp carries into luts
[-debug_log]	show debug message
[-property_opt_only]	Do targeted optimizations on tagged cells
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Description

Optimizes a design netlist for the target part. Optimization can provide improvements to synthesized netlists from third-party tools, or for netlists that may not have been optimized during synthesis.

Run this command after synthesis but prior to implementation to optimize the design and simplify the netlist before placing and routing the design. Assign the `DONT_TOUCH` property to prevent the optimization of individual cells.

> **TIP:** *To see what actions `opt_design` is taking in optimizing your design, you can use the `-verbose` option to get a more detailed transcript of the process. This can help you in understanding and debugging some of the changes made to your design.*

The `opt_design` command performs the following optimizations by default:

- Retarget

- Constant Propagation

- Sweep

- Global Buffer (BUFG) optimizations

- DSP Register optimizations

- Shift-Register Logic optimizations

- Block RAM Power optimizations

- Implement MIG cores

- Implement Debug cores

> **IMPORTANT!** *Using command-line options for specific optimizations results in `opt_design` performing only the specified optimizations and disabling all others, even the ones that are usually performed by default.*

To perform LUT Remapping, you must specify `-remap`.

To perform area-based re-synthesis, you must specify `-resynth_area`, or `-directive ExploreArea`.

To perform sequential area-based re-synthesis, you must specify `-resynth_seq_area`, or `-directive ExploreSequentialArea`.

Arguments

`-retarget` - (Optional) Retarget one type of block to another when retargetting the design from one device family to another. For example, retarget instantiated MUXCY or XORCY components into a CARRY4 block; or retarget DCM to MMCM. The retarget optimization also absorbs inverters into downstream logic where possible.

Note: The `-retarget` argument is optional, as are the other optimizations. However this optimization is run by default unless explicitly overridden by another optimization.

`-propconst` - (Optional) Propagate constant inputs through the circuit, resulting in a simplified netlist. Propagation of constants can eliminate redundant combinational logic from the netlist.

`-sweep` - (Optional) Remove unnecessary logic, removing loadless cells and nets. Additional optimization can be performed such as tie-off, register replication driving multiple OBUF, IBUF/OBUF insertion, retargeting dual port RAM to single port RAM, and optimizing IDELAYCTRL by grouping IO.

`-bram_power_opt` - (Optional) Enables power optimization on Block RAM cells. Changes the WRITE_MODE on unread ports of true dual-port RAMs to NO_CHANGE, and applies intelligent clock gating to Block RAM outputs.

Note: Specific BRAM cells can be excluded from this optimization using the `set_power_opt` command.

`-remap` - (Optional) Remap the design to combine multiple LUTs into a single LUT to reduce the depth of the logic.

`-aggressive_remap` - (Optional) Similar to the `-remap` option to reduce LUT logic depth but more exhaustive. This may reduce more LUT levels at the expense of longer optimization runtime.

`-resynth_area` - (Optional) Perform re-synthesis in area mode to reduce the number of LUTs.

`-resynth_seq_area` - (Optional) Perform re-synthesis to reduce both combinational and sequential logic. Performs a superset of the optimization provided by `-resynth_area`.

`-resynth_remap` - (Optional) Remap the design to improve the critical paths in timing-driven mode by performing re-synthesis to reduce the depth of logic. This timing-based approach will replicate LUT's with fanout and collapse smaller LUT's into bigger functions at the expense of longer optimization runtime..

`-directive <arg>` - (Optional) Direct the mode of optimization with specific design objectives. Only one directive can be specified for a single `opt_design` command, and values are case-sensitive. Supported values include:

- `Explore` - Run multiple passes of optimization to improve results.

- `ExploreArea` - Run multiple passes of optimization, with an emphasis on reducing area.

Send Feedback

- `ExploreWithRemap` - Similar to ExploreArea but adds the remap optimization to compress logic levels.

- `ExploreSequentialArea` - Run multiple passes of optimization, with an emphasis on reducing registers and related combinational logic.

- `AddRemap` - Run the default optimization, and include LUT remapping to reduce logic levels.

- `NoBramPowerOpt` - Runs `opt_design` without the default BRAM power optimization.

- `RuntimeOptimized` - Run the fewest iterations, trading optimization results for faster runtime.

- `RQS` - Instructs `opt_design` to select the `opt_design` directive specified by the `report_qor_suggestion` strategy suggestion. Requires an RQS file with a strategy suggestion to be read in prior to calling this directive. See `report_qor_suggestions - help` for more details.

- `Default` - Run the default optimization.

Refer to the *Vivado Design Suite User Guide: Implementation (UG904)* for more information on the effects of each directive.

Note: The `-directive` option controls the overall optimization strategy, and is not compatible with any specific optimization options. It can only be used with `-debug_log`, `-quiet` and `-verbose`.

`-muxf_remap` - (Optional) Convert MUXFs to LUT3s opportunistically when it can potentially improve route-ability of the design without affecting timing.

`-hier_fanout_limit <arg>` - (Optional) Net drivers with fanout greater than the specified limit (<arg>) will be replicated according to the logical hierarchy. For each hierarchical instance driven by the high-fanout net, if the fanout within the hierarchy is greater than the specified limit, then the net within the hierarchy is driven by a replica of the driver of the high-fanout net. The minimum value for `hier_fanout_limit` is 512.

`-bufg_opt` - (Optional) Perform various optimizations related to global buffers (BUFG/BUFGCE): Insert a buffer on unbuffered clock nets (fanout > 30). Insert BUFGs on high fanout nets where the control set pin count on the net exceeds the tool threshold. Perform load-splitting when a high-fanout net drives both combinational and sequential logic: the combinational portion bypasses the BUFG because the added delay is too large.

> **TIP:** The `phys_opt_design` command can be used to optimize the combinational portion, while the sequential portion can be driven by BUFG.

`-mbufg_opt` - (Optional) MBUFG optimization transforms parallel clock buffers driven by a common driver, or clock modifying blocks (CMB) such as MMCM, DPLL, or XPLL, to MBUFG.

`-shift_register_opt` - (Optional) For high-fanout nets originating from an SRL, add output register pipeline stages to improve timing prior to the high-fanout net replication. This optimization is performed as part of the default optimization.

`-dsp_register_opt` - (Optional) Pull registers out of a DSP, or push registers into a DSP, to improve timing and resource utilization as needed. This optimization is performed as part of the default optimization.

`-srl_remap_modes <args>` - (Optional) Many designs have timing closure issues related to deep SRLs, and long register chains. This option provides both an automatic rule to trigger remapping between FF chains and SRLs, and also a manual rule to specify when to convert a FF chain to an SRL, or an SRL to a FF chain. Automatic remapping starts with the longest FF chains and converts those to SRLs when needed to reduce FF utilization, and takes the shallowest SRLs and converts those to FF chains when needed to reduce LUT utilization. Manual remapping takes FF chains in excess of specified lengths and remaps them to SRLs, and takes SRLs with specified depths and remaps them to FF chains. The automatic and manual modes cannot both be used in the same opt_design command pass. The following shows the syntax for the automatic and manual rules:

- Automatic:
  - `-srl_remap_modes {{target_ff_util <util> target_lutram_util <util>}}`: Where <util> is a number from 0 to 100 expressed as a percentage of utilization. When either FF or LUTs exceed the specified utilization, the tool remaps as needed. When both utilization rates are exceeded neither is remapped.

- Manual:
  - `-srl_remap_modes {{min_depth_ffs_to_srl <val>} {max_depth_srl_to_ffs <val>}}`: Where <val> specifies the depth of an SRL or length of a FF chain. When a FF chain is greater than the specified min value, the chain is remapped to an SRL with the required depth. If an SRL has depth less than or equal to the specified max value, the SRL is remapped to a FF chain.

`-control_set_merge` - (Optional) Reduce the drivers of logically-equivalent control signals to a single driver. This is like a reverse fanout replication, and results in nets that are better suited for module-based replication.

`-merge_equivalent_drivers` - (Optional) This option merges equivalent drivers on both control signals and non-control signals, to reduce logic in the design.

`-carry_remap` - (Optional) Remap carry signals into LUTs.

`-debug_log` - (Optional) Generate log file for debugging purposes.

`-property_opt_only` - (Optional) Indicates that the `opt_design` command should be driven only by properties on objects in the design, and not by the options defined as part of the command syntax. Refer to the *Vivado Design Suite Properties Reference Guide* (UG912) for more information on specific properties. The following properties trigger property-only optimizations: CARRY_REMAP, CONTROL_SET_REMAP, EQUIVALENT_DRIVER_OPT, LUT_REMAP, MUXF_REMAP, REG_TO_SRL, SRL_STAGES_TO_REG_INPUT, SRL_STAGES_TO_REG_OUTPUT, SRL_TO_REG.

> **TIP:** *When* `-property_opt_only` *is specified, the tool will return an error if additional optimizations have been specified in the* `opt_design` *command.*

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command. This option displays detailed information about the logic that is affected by each optimization.

Examples

The following example performs all four default optimizations: retarget, constant propagation, sweep, and BRAM power optimization. The command returns detailed results with the `-verbose` switch:

```
opt_design -verbose
```

This example excludes specific BRAM cells from power optimization using the `set_power_opt` command, and then runs `opt_design` with the four default optimizations:

```
set_power_opt -exclude_cells [get_cells \
    -filter {PRIMITIVE_TYPE =~ BMEM.*.*} \
    -of_objects [get_pins -leaf -filter {DIRECTION == IN} \
    -of_objects [get_nets -of_objects [get_pins clock/bufgctrl_clk_mld/O]]]]
opt_design
```

The following example performs the sweep and retarget optimizations:

```
opt_design -sweep -retarget
```

Note: Because `-sweep` and `-retarget` are expressly enabled in the prior example, `-propconst` optimization and `-bram_power_opt` are implicitly disabled.

The following example directs the `opt_design` command to use various algorithms to achieve potentially better results:

```
opt_design -directive Explore
```

The following example directs the `opt_design` command to use various algorithms to achieve potentially better results, while focusing on area reduction:

```
opt_design -directive ExploreArea
```

Send Feedback

This example uses the `-srl_remap_modes` argument with the automatic option to convert FF chains into SRLs when FF utilization is greater than 20%, or SRLs into FF chains when SRL utilization is over 50%:

```
-srl_remap_modes {{target_ff_util 20 target_lutram_util 50}}
```

TIP: *In this case, the srl_remap is only performed if either specified utilization rate is exceeded, but NOT if both are exceeded.*

The following example uses the `-srl_remap_modes` argument with the manual option to convert FF chains with length greater than 5 to SRLs, and SRLs with depth less than or equal to 8 to FF chains:

```
-srl_remap_modes {{min_depth_ffs_to_srl 5}{max_depth_srl_to_ffs 8}}
```

See Also

- phys_opt_design
- place_design
- power_opt_design
- route_design
- set_power_opt
- synth_design

Send Feedback

pause_hw_hbm_amon

Pause Activity Monitor runs for the specified hardware HBM(s)

Syntax

```
pause_hw_hbm_amon [-quiet] [-verbose] <hw_objects>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hw_objects>	hardware objects

Categories

[Hardware](#)

Description

The `pause_hw_hbm_amon` command pauses a running HBM activity monitor in the Vivado hardware manager that has been previously started using the `run_hw_hbm_amon` command.

This command returns nothing when successful, or returns an error when it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_objects> - (Required) Defines the HBM activity monitor objects (hw_hbms) to pause.

Examples

The following example pauses the HBM activity monitor for the associated HBM core:

```
pause_hw_hbm_amon [get_hw_hbms *HBM_2]
```

See Also

- add_hw_hbm_pc
- commit_hw_hbm
- current_hw_device
- current_hw_target
- get_hw_hbms
- refresh_hw_hbm
- remove_hw_hbm_pc
- resume_hw_hbm_amon
- run_hw_hbm_amon
- stop_hw_hbm_amon

phys_opt_design

Optimize the current placed netlist.

Syntax

```
phys_opt_design [-fanout_opt] [-placement_opt] [-routing_opt]
    [-slr_crossing_opt] [-rewire] [-insert_negative_edge_ffs]
    [-critical_cell_opt] [-dsp_register_opt] [-bram_register_opt]
    [-uram_register_opt] [-bram_enable_opt] [-shift_register_opt]
    [-hold_fix] [-aggressive_hold_fix] [-retime]
    [-force_replication_on_nets <args>] [-directive <arg>]
    [-critical_pin_opt] [-clock_opt] [-path_groups <args>] [-tns_cleanup]
    [-sll_reg_hold_fix] [-quiet] [-verbose]
```

Usage

Name	Description
[-fanout_opt]	Do cell-duplication based optimization on high-fanout timing critical nets
[-placement_opt]	Do placement based optimization on timing critical nets
[-routing_opt]	Do routing based optimization on timing critical nets
[-slr_crossing_opt]	Do placement optimization of SLR-crossing timing critical nets
[-rewire]	Do rewiring optimization
[-insert_negative_edge_ffs]	Insert negative edge triggered FFs for hold optimization
[-critical_cell_opt]	Do cell-duplication based optimization on timing critical nets
[-dsp_register_opt]	Do DSP register optimization
[-bram_register_opt]	Do BRAM register optimization
[-uram_register_opt]	Do UltraRAM register optimization
[-bram_enable_opt]	Do BRAM enable optimization
[-shift_register_opt]	Do Shift register optimization
[-hold_fix]	Attempt to improve slack of high hold violators
[-aggressive_hold_fix]	Attempt to aggressively improve slack of high hold violators
[-retime]	Do retiming optimization
[-force_replication_on_nets]	Force replication optimization on nets
[-directive]	Mode of behavior (directive) for this command. Please refer to Arguments section of this help for values for this option Default: Default
[-critical_pin_opt]	Do pin-swapping based optimization on timing critical nets
[-clock_opt]	Do clock skew optimization in post-route optimization
[-path_groups]	Work only on specified path groups
[-tns_cleanup]	Work on all nets in the design that meet criteria for the specified optimizations to improve design tns
[-sll_reg_hold_fix]	Do hold fixing on SLL Tx-Rx paths
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Tools](#)

Description

Performs timing-driven optimization on the negative-slack paths of a design. A path should have negative slack near the worst negative slack (WNS) to be considered for optimization. Optimization will not be performed on designs without negative slack.

This optional command can be run for post-place or post-route optimizations.

Note: Because physical optimization requires timing data that is only available after placement, the command cannot be run prior to placement. However, the `write_iphys_opt_tcl` and `read_iphys_opt_tcl` commands let you write out the physical optimizations performed on the post-placed design, and then apply those optimizations to the design netlist prior to placement. Refer to the *Vivado Design Suite User Guide: Implementation* (UG904) for more information on interactive physical optimization.

Post-place `phys_opt_design` performs the following optimizations by default:

- high-fanout optimization

- placement-based optimization of critical paths

- rewire

- critical-cell optimization

- DSP register optimization

- BRAM register optimization

- URAM register optimization

- a final fanout optimization

> **TIP:** *Using command-line options for specific optimizations results in `phys_opt_design` performing only the specified optimizations and disabling all others, even the ones that are usually performed by default.*

Post-route `phys_opt_design` performs the following optimizations by default:

- placement-based optimization of critical paths

- routing optimization

- rewire

- critical-cell optimization

Physical optimizations involve replication, re-timing, hold fixing, and placement improvement. The `phys_opt_design` command automatically performs all necessary netlist and placement changes.

To perform re-timing you must specify the `-retime` option, or the `-directive AddRetime` option.

To perform hold fixing you must specify the `-hold_fix` option, or the `-directive ExploreWithHoldFix` option.

If the `phys_opt_design` command is used iteratively, the subsequent run optimizes the results of the prior run.

> **TIP:** *The `phys_opt_design` can be multi-threaded to speed the process. Refer to the `set_param` command for more information on setting the `general.maxThreads` parameter.*

The command reports each net processed, a summary of any optimizations performed, and the WNS before and after optimization. Replicated objects are named by appending `_replica` to the original object name, followed by the replicated object count.

Arguments

`-fanout_opt` - (Optional) Performs delay-driven optimization on high-fanout timing critical nets, by replicating drivers to reduce delay.

Note: The `-fanout_opt` argument is optional, as are the other optimizations. However this optimization is run by default unless explicitly overridden by another optimization.

`-placement_opt` - (Optional) Moves cells to reduce delay on timing-critical nets.

`-routing_opt` - (Optional) Performs routing optimization on timing-critical nets to reduce delay.

`-slr_crossing_opt` - (Optional) Performs post-place or post-route optimizations to improve the path delay of inter-SLR connections. The optimization adjusts the locations of the driver, load, or both along the SLR crossing after potential replication. For use with UltraScale and UltraScale+ devices.

`-rewire` - (Optional) Refactors logic cones to reduce logic levels and reduce delay on critical signals.

`-insert_negative_edge_ffs` - (Optional) Inserts negative edge triggered FFs for hold optimization.

`-critical_cell_opt` - (Optional) Replicates cells on timing critical nets to reduce delays.

`-dsp_register_opt` - (Optional) Improves critical path delay by moving registers from slices to DSP blocks, or from DSP blocks to slices.

`-bram_register_opt` - (Optional) Improves critical path delay by moving registers from slices to block RAMs, or from block RAMs to slices.

`-uram_register_opt` - (Optional) Improves critical path delay by moving registers across the URAM cell boundary, from slices into URAMs, or from URAMs to slices.

`-bram_enable_opt` - (Optional) The block RAM enable optimization improves timing on critical paths involving power-optimized block RAMs. Pre-placement block RAM power optimization restructures the logic driving block RAM read and write enable inputs to reduce dynamic power consumption. After placement the restructured logic may become timing-critical. The block RAM enable optimization reverses the enable-logic optimization to improve the slack on the critical enable-logic paths.

`-shift_register_opt` - (Optional) Performs shift register optimization to improve timing on negative slack paths between shift register cells (SRLs) and other logic cells. If there are timing violations to or from shift register cells (SRL16E or SRLC32E), the optimization extracts a register from the beginning or end of the SRL register chain and places it into the logic fabric to improve timing. The optimization shortens the wire length of the original critical path. Refer to the *Vivado Design Suite User Guide: Implementation (UG904)* for more information.

`-hold_fix` - (Optional) Performs optimizations to insert data path delay to fix hold time violations.

`-aggressive_hold_fix` - (Optional) Performs optimizations to insert data path delay to fix hold time violations. Considers significantly more hold violations than the standard hold fix algorithm.

`-retime` - (Optional) Re-time registers forward through combinational logic to balance path delays.

`-force_replication_on_nets` <args> - (Optional) Forces the drivers of the specified net objects to be replicated, regardless of timing slack. This option requires the use of the `get_nets` command to specify net objects. Replication is based on load placements and requires manual analysis to determine if replication is sufficient. If further replication is required, nets can be replicated repeatedly by successive commands.

`-directive` <arg> - (Optional) Directs the mode of physical optimization with specific design objectives. Only one directive can be specified for a single `phys_opt_design` command, and values are case-sensitive. Supported values include:

- `Explore` - Run different algorithms in multiple passes of optimization, including replication for very high fanout nets.

- `ExploreWithHoldFix` - Run different algorithms in multiple passes of optimization, including hold violation fixing and replication for very high fanout nets.

- `ExploreWithAggressiveHoldFix` - Run different algorithms in multiple passes of optimization, including aggressive hold violation fixing and replication for very high fanout nets.

- `AggressiveExplore` - Similar to Explore but with different optimization algorithms and more aggressive goals.

- `AlternateReplication` - Use different algorithms for performing critical cell replication.

- `AggressiveFanoutOpt` - Uses different algorithms for fanout-related optimizations with more aggressive goals.

- `AlternateFlowWithRetiming` - Perform more aggressive replication and DSP and BRAM optimizations, and enable register re-timing.

- `AddRetime` - Performs the default `phys_opt_design` flow and adds re-timing.

- `RuntimeOptimized` - Provides a reduced set of physical optimizations with the shortest runtime. Use this directive when compile time reduction is more important than design performance. RuntimeOptimized includes fanout_opt, critical_cell_opt, placement_opt, and bram_enable_opt.

- `Default` - Run `phys_opt_design` with default settings.

Refer to the *Vivado Design Suite User Guide: Implementation (UG904)* for more information on the effects of each directive.

Note: The `-directive` option controls the overall optimization strategy, and is not compatible with any specific optimization options. It can only be used with `-quiet` and `-verbose`.

`-critical_pin_opt` - For LUT inputs, this optimization performs remapping of logical pins to physical pins, also known as pin-swapping, to improve critical path timing. A critical path traversing a logical pin that has been mapped to a slow physical pin, such as A1 or A2, is reassigned to a faster physical pin, such as A6 or A5 if it improves timing.

Note: A cell with a LOCK_PINS property will be skipped and the cell will retain the pin mapping specified by LOCK_PINS. Logical-to-physical pin mapping can be obtained with `get_site_pins` for a selected logical pin object.

`-clock_opt` - (Optional) Perform clock skew optimization during post-route optimization. Insert global clock buffers to delay destination clocks to improve setup on critical paths.

`-path_groups` <args> - (Optional) Perform optimizations on the specified path groups only.

`-tns_cleanup` - (Optional) This option can be used with the `-slr_crossing_opt` option. This performs a Total Negative Slack (TNS) cleanup, which supports optimizing SLR crossing paths while allowing some slack degradation on other paths, as long as the overall WNS does not degrade. This is applicable to UltraScale and UltraScale+ devices.

`-sll_reg_hold_fix` - (Optional) Performs SLL register hold fix optimization. You can use this option when the router is having trouble resolving hold violations on SLR crossing paths.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example performs a physical optimization of the current post-placement design, and then writes the iphys_opt Tcl script for use before placement:

```
phys_opt_design
write_iphys_opt_tcl C:/Data/my_iphys_opt.tcl
```

This example sets the LOCK_PINS property on the specified cell, then performs physical optimizations including register re-timing, optimization of registers across DSP blocks and block RAMs, and pin swapping (excluding the locked pins) to improve timing:

```
set_property LOCK_PINS {I3:A1 I2:A4} [get_cell cpuEngine/qmem_dack_reg_i_1]
phys_opt_design -retime -dsp_register_opt -bram_register_opt \
    -critical_pin_opt
```

This example directs `phys_opt_design` to run more iterations, with hold violation fixing, to achieve potentially better results:

```
phys_opt_design -directive ExploreWithHoldFix
```

This example directs `phys_opt_design` to consider more nets for replication:

```
phys_opt_design -directive AggressiveFanoutOpt
```

See Also

- get_pins
- get_site_pins
- opt_design
- place_design
- power_opt_design
- read_iphys_opt_tcl
- route_design
- write_iphys_opt_tcl

Send Feedback

place_cell

Move or place one or more instances to new locations. Sites and cells are required to be listed in the right order and there should be same number of sites as number of cells.

Syntax

```
place_cell [-quiet] [-verbose] <cell_site_list>...
```

Usage

Name	Description
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<cell_site_list>`	a list of cells and sites in the interleaved order

Categories

[Floorplan](#)

Description

Places cells onto device resources of the target part. Cells can be placed onto specific BEL sites (e.g. `SLICE_X49Y60/A6LUT`), or into available SLICE resources (e.g. `SLICE_X49Y60`). If you specify the SLICE but not the BEL the tool will determine an appropriate BEL within the specified SLICE if one is available.

When placing a cell onto a specified site, the site must not be currently occupied, or an error will be returned: `Cannot set site and bel property of instances. Site SLICE_X49Y61 is already occupied.`

You can test if a site is occupied by querying the IS_OCCUPIED property of a BEL site:

```
get_property IS_OCCUPIED [get_bels SLICE_X48Y60/D6LUT]
```

Note: The IS_OCCUPIED property of a SLICE only tells you if some of the BELs within the SLICE are occupied; not whether or not the SLICE is fully occupied.

This command can be used to place cells, or to move placed cells from one site on the device to another site. The command syntax is the same for placing an unplaced cell, or moving a placed cell.

Send Feedback

When moving a placed cell, if you specify only the SLICE for the site, the tool will attempt to place the cell onto the same BEL site in the new SLICE as it currently is placed. For instance moving a cell from the B6LUT, by specifying a new SLICE, will cause the tool to attempt to place the cell onto the B6LUT in the new SLICE. If this BEL site is currently occupied, an error is returned.

Note: This command returns nothing if successful, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<cell_site_list> - (Required) Specifies a list of cells and sites as {<cell_name> <site> }. The cell name is listed first, followed the BEL site or SLICE to place the cell onto. If the site is specified as a SLICE, the tool will select an available BEL within the SLICE. Multiple cells can be placed onto multiple sites by repeating the cell/site pair multiple times as needed:

```
{<cell_name1> <site1> <cell_name2> <site2> \
<cell_name3> <site3>... <cell_nameN> <siteN> }
```

Examples

The following example places the specified cell onto the specified BEL site:

```
place_cell div_cntr_reg_inferredi_4810_15889 SLICE_X49Y60/D6LUT
```

The following example places the specified cell into the specified SLICE:

```
place_cell div_cntr_reg_inferredi_4810_15889 SLICE_X49Y61
```

Note: The tool will select an appropriate BEL site if one is available. If no BEL is available, and error will be returned.

The following example places multiple cells onto multiple sites:

```
place_cell  { \
   cpuEngine/cpu_iwb_adr_o/buffer_fifo/i_4810_17734 SLICE_X49Y60/A6LUT \
   cpuEngine/or1200_cpu/or1200_mult_mac/i_4775_15857 SLICE_X49Y60/B6LUT \
   cpuEngine/cpu_iwb_adr_o/buffer_fifo/xlnx_opt_LUT_i_4810_18807_2 \
   SLICE_X49Y60/C6LUT }
```

See Also

- create_cell
- remove_cell
- unplace_cell

place_design

Automatically place ports and leaf-level instances

Syntax

```
place_design [-directive <arg>] [-auto_directive_index <arg>]
    [-no_timing_driven] [-timing_summary] [-unplace] [-post_place_opt]
    [-no_psip] [-no_bufg_opt] [-quiet] [-verbose]
```

Usage

Name	Description
[-directive]	Mode of behavior (directive) for this command. Please refer to Arguments section of this help for values for this option. Default: Default
[-auto_directive_index]	Index (starting from 1) of directive among sorted list of directives based on descending order of probability to improve timing Default: 1
[-no_timing_driven]	Do not run in timing driven mode
[-timing_summary]	Enable accurate post-placement timing summary.
[-unplace]	Unplace all the instances which are not locked by Constraints.
[-post_place_opt]	Run only the post commit optimizer
[-no_psip]	Disable PSIP (Physical Synthesis In Placer) optimization during placement.
[-no_bufg_opt]	Disable global buffer insertion during placement
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Tools

Description

Place the specified ports and logic cells in the current design, or all ports and logic cells, onto device resources on the target part. The tool optimizes placement to minimize negative timing slack and reduce overall wire length, while also attempting to spread out placement to reduce routing congestion.

Placement is one step of the complete design implementation process, which can be run automatically through the use of the `launch_runs` command when running the Vivado tools in Project Mode.

In Non-Project Mode, the implementation process must be run manually with the individual commands: `opt_design`, `place_design`, `phys_opt_design`, `power_opt_design`, and `route_design`. Refer to the *Vivado Design Suite User Guide: Design Flows Overview* (UG892) for a complete description of Project Mode and Non-Project Mode.

Both placement and routing can be completed incrementally, based on prior results stored in a Design Checkpoint file (DCP), using the incremental implementation flow. Refer to the `read_checkpoint` command, or to *Vivado Design Suite User Guide: Implementation* (UG904) for more information on incremental place and route.

> **TIP:** *The* `place_design` *can be multi-threaded to speed the process. Refer to the* `set_param` *command for more information on setting the* `general.maxThreads` *parameter.*

You can also manually place some elements of the design using `place_ports`, or by setting LOC properties on the cell, and then automatically place the remainder of the design using `place_design`.

This command requires an open synthesized design, and it is recommended that you run the `opt_design` command prior to running `place_design` to avoid placing a suboptimal netlist.

Arguments

`-directive <arg>` - (Optional) Direct placement to achieve specific design objectives. Only one directive can be specified for a single `place_design` command, and values are case-sensitive. Supported values include:

- `Explore` - Increased placer effort in detail placement and post-placement optimization .

- `EarlyBlockPlacement` - Timing-driven placement of RAM and DSP blocks. The RAM and DSP block locations are finalized early in the placement process and are used as anchors to place the remaining logic.

- `WLDrivenBlockPlacement` - Wire length-driven placement of RAM and DSP blocks. Override timing-driven placement by directing the Vivado placer to minimize the distance of connections to and from blocks.

- `ExtraNetDelay_high` - Increases estimated delay of high fanout and long-distance nets. Three levels of pessimism are supported: high, medium, and low. `ExtraNetDelay_high` applies the highest level of pessimism.

- `ExtraNetDelay_low` - Increases estimated delay of high fanout and long-distance nets. Three levels of pessimism are supported: high, medium, and low. `ExtraNetDelay_low` applies the lowest level of pessimism.

- `AltSpreadLogic_high` - Spreads logic throughout the device to avoid creating congested regions. Three levels are supported: high, medium, and low. AltSpreadLogic_high achieves the highest level of spreading.

- `AltSpreadLogic_medium` - Spreads logic throughout the device to avoid creating congested regions. Three levels are supported: high, medium, and low. AltSpreadLogic_medium achieves a medium level of spreading compared to low and high.

- `AltSpreadLogic_low` - Spreads logic throughout the device to avoid creating congested regions. Three levels are supported: high, medium, and low. AltSpreadLogic_low achieves the lowest level of spreading.

- `ExtraPostPlacementOpt` - Increased placer effort in post-placement optimization.

- `ExtraTimingOpt` - Use an alternate algorithm for timing-driven placement with greater effort for timing.

- `SSI_SpreadLogic_high` - Distribute logic across SLRs. `SSI_SpreadLogic_high` achieves the highest level of distribution.

- `SSI_SpreadLogic_low` - Distribute logic across SLRs. `SSI_SpreadLogic_low` achieves a minimum level of logic distribution, while reducing placement runtime.

- `SSI_SpreadSLLs` - Partition across SLRs and allocate extra area for regions of higher connectivity.

- `SSI_BalanceSLLs` - Partition across SLRs while attempting to balance SLLs between SLRs.

- `SSI_BalanceSLRs` - Partition across SLRs to balance number of cells between SLRs.

- `SSI_HighUtilSLRs` - Direct the placer to attempt to place logic closer together in each SLR.

- `RuntimeOptimized` - Run fewest iterations, trade higher design performance for faster runtime.

- `Quick` - Absolute, fastest runtime, non-timing-driven, performs the minimum required placement for a legal design.

- `RQS` - Instructs `place_design` to select the `place_design` directive specified by the `report_qor_suggestion` strategy suggestion. Requires an RQS file with a strategy suggestion to be read in prior to calling this directive. See `report_qor_suggestions -help` for more details.

- `Auto` - Instructs `place_design` to use machine learning to select the best predicted directives. The directive selected is based on the `-auto_directive_index` switch. 1 is the directive with the best prediction.

- `Default` - Run `place_design` with default settings.

> **IMPORTANT!** *The* `-directive` *option controls the overall placement strategy, and is not compatible with some* `place_design` *options. It can be used with* `-no_psip`, `-no_bufg_opt`, `-quiet` *and* `-verbose`. *Only the* `Explore`, `Quick`, *and* `Default` *directives are compatible with high reuse designs and the incremental implementation flow as defined by* `read_checkpoint -incremental`. *Refer to the Vivado Design Suite User Guide: Implementation (UG904) for more information on placement strategies and the use of the* `-directive` *option.*

`-auto_directive_index` - (Optional) When specified with -directive Auto, specifies which predicted directive should be run. 1 is the best predicted directive. Typically a number of 5 or less should be specified with a recommended value up to 3. Default value is 1.

`-no_timing_driven` - (Optional) Disables the default timing driven placement algorithm. This results in a faster placement based on wire lengths, but ignores any timing constraints during the placement process.

`-timing_summary` - (Optional) Report the post-placement worst negative slack (WNS) using results from static timing analysis. The WNS value is identical to that of `report_timing_summary` when run on the post-placement design. By default the placer reports an estimated WNS based on incremental placement updates during the design implementation. The `-timing_summary` option incurs additional runtime to run a full timing analysis.

`-unplace` - (Optional) Unplace all the instances which are not locked by constraints. Cells with fixed placement (IS_LOC_FIXED set to TRUE), are not affected.

> **TIP:** *Use the* `set_property` *to change IS_LOC_FIXED to FALSE prior to unplacing fixed cells.*

`-post_place_opt` - (Optional) Run optimization after placement to improve critical path timing at the expense of additional placement and routing runtime. This optimization can be run at any stage after placement. The optimization examines the worst case timing paths and tries to improve placement to reduce delay.

> **TIP:** *Any placement changes will result in unrouted connections, so* `route_design` *will need to be run after* `-post_place_opt`.

`-no_psip` - (Optional) Disable PSIP (Physical Synthesis In Placer) optimization during placement. By default, to improve delay the Vivado placer performs optimizations such as replicating drivers of high-fanout nets and drivers of loads that are far-apart. This option disables those optimizations.

`-no_bufg_opt` - (Optional) By default, global buffers are inserted during placement to drive high-fanout nets. This option disables global buffer insertion to reduce the number of routing resources consumed by high fanout nets that are not timing-critical.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Send Feedback

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example places the current design, runs optimization, routes the design, runs post placement optimization, and then reroutes the design to cleanup any unconnected nets as a result of post placement optimization:

```
place_design
phys_opt_design
route_design
place_design -post_place_opt
phys_opt_design
route_design
```

The following example directs the Vivado placer to try different placement algorithms to achieve a better placement result:

```
place_design -directive Explore
```

The following example directs the Vivado placer to use the Machine Learning based predictor to select the top directive:

```
place_design -directive Auto
```

The following example directs the Vivado placer to use the Machine Learning based predictor to select the second best predicted directive:

```
place_design -directive Auto -auto_directive_index 2
```

This example unplaces the current design:

```
place_design -unplace
```

See Also

- launch_runs
- opt_design
- place_ports
- phys_opt_design
- power_opt_design
- read_checkpoint
- route_design
- set_property

place_ports

Automatically place a set of ports

Syntax

```
place_ports [-skip_unconnected_ports] [-check_only] [-iobank <args>]
      [-quiet] [-verbose] [<ports>...]
```

Usage

Name	Description
[-skip_unconnected_ports]	Do not place unconnected ports
[-check_only]	Only check IO/Clock placement DRCs
[-iobank]	Limit placement to the following banks
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<ports>]	Ports to place (if omitted, all ports will be placed). If the arguments are interleaved objects of ports and package pins, then manual placement is performed

Categories

PinPlanning

Description

Assign ports to the pins of the Xilinx FPGA package, by automatically or manually placing ports.

- Automatically places ports on an available I/O or clocking site, or into the specified I/O banks.

- Manually assigns ports to the specified package_pin when both the port and pin are specified.

The `place_ports` command will not replace ports that are currently placed by the user, or ports that are placed and fixed.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

`-skip_unconnected_ports` - (Optional) Do not place unconnected ports.

`-check_only` - (Optional) Run the clock placer DRCs, which are also available as PLCK checks in the `report_drc` command. This option does not result in ports being placed, only checked for valid placement.

`-iobank` <args> - (Optional) Place the specified ports into the listed IO bank objects. IO bank objects are returned by the `get_iobanks` command.

Note: Limiting port placement to specific IO banks will result in a placement error if there are insufficient placement sites for the number of ports being placed.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<ports> - (Optional) The names of the ports to be placed.

- If no <ports> are specified, all unplaced ports will be placed.

- Ports can be specified in <port_name> and <package_pin> pairs to manually assign ports to the specified package_pin. Multiple port_name package_pin pairs can be specified as follows:

```
{<port_name1> <package_pin1> <port_name2> <package_pin2>
<port_name2> <package_pin2>...<port_nameN> <package_pinN> }
```

Note: If previously placed ports are specified, or included in the list of ports to place, the Vivado tool will not replace or move those ports.

Examples

The following example places the port objects returned by the `get_ports` command, onto I/O bank 13 of the device, as returned by `get_iobanks`:

```
place_ports -iobank [get_iobanks 13] [get_ports DataOut_pad_1_o]
```

The follow example uses port_name package_pin pairs to manually place multiple ports:

```
place_ports {LEDS_n[2] AA11 LEDS_n[3] AA10 LEDS_n[0] Y11 LEDS_n[1] Y10}
```

The following example places all input ports onto I/O banks 12, 13, 14 and 15 of the device:

```
place_ports -iobank [get_iobanks {12 13 14 15}] [all_inputs]
```

See Also

- create_port
- get_iobanks

Send Feedback

- [make_diff_pair_ports](#)
- [remove_port](#)

platform_verify

TCL task to verify platform by comparing files at the bit level

Syntax

```
platform_verify -reference <arg> [-untrusted <arg>] [-in_memory]
    [-cell <arg>] [-quiet] [-verbose]
```

Usage

Name	Description
`-reference`	full path to trusted reference DCP file
`[-untrusted]`	full path to untrusted DCP file
`[-in_memory]`	use in-memory design for comparison, when -untrusted option is not specified
`[-cell]`	Name of reconfigurable cell which will be verified
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution

Categories

Description

This command takes two design checkpoint files (DCP) and compares the binary content of the files, comparing them at the bit-level.

Arguments

`-reference <arg>` - (Required) Specifies the known good DCP as a reference to compare.

`-untrusted <arg>` - (Optional) Specifies an unqualified DCP as the file to be verified.

> ⭐ **IMPORTANT!** `-untrusted` and `-in_memory` are both optional. However, at least one of these options must be specified to indicate the untrusted design source.

`-in_memory` - (Optional) Use the `current_design` for comparison, when the `-untrusted` option is not specified.

`-cell <arg>` - (Optional) Instructs the tool to compare the specified cell in the two DCP files.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following command compares the two design checkpoints:

```
platform_verify -reference C:/Data/design1/routed.dcp -untrusted C:/Data/
design2/routed.dcp
```

The following command compares the current design against the specified design checkpoint in memory:

```
platform_verify -reference C:/Data/design1/routed.dcp -in_memory
```

See Also

- current_design
- read_checkpoint
- write_checkpoint

power_opt_design

Optimize dynamic power using intelligent clock gating

Syntax

```
power_opt_design [-quiet] [-verbose]
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Power

Description

Optimizes the dynamic power consumption of the design by changing clock gating to take advantage of clock enable on a flop. Clock gating optimizations are automatically performed on the entire design to improve power consumption while making no changes to the existing logic or the clocks that would alter the behavior of the design.

You can configure the power optimization to include or exclude specific cells using the `set_power_opt` command.

Note: Block RAM power optimizations are performed by default with the `opt_design` command. You can disable BRAM optimization by changing the defaults of `opt_design`, or by excluding specific cells from optimization using the `set_power_opt` command.

You can also use the `read_saif` command prior to optimization, and `power_opt_design` will consider the activity data while optimizing the design.

You can run power optimization after synthesis, or after placement. When run before placement, this command optimizes the design to save power. When run after placement, this command optimizes the design to save power while preserving timing. Running after placement limits the optimizations available to the `power_opt_design` command. To achieve the best results, the command should be run prior to placement.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example performs power optimization of the open design:

```
power_opt_design
```

This example optimizes the design, excluding the BRAM power optimization by specifying the optimizations to run, and then runs power optimization on the design:

```
opt_design -retarget -propconst -sweep
power_opt_design
```

See Also

- opt_design
- phys_opt_design
- read_saif
- report_power
- report_power_opt
- set_power_opt

pr_recombine

Re-establish a parent cell as a Reconfigurable Partition while removing a lower-level Reconfigurable Partition when using the Hierarchical Dynamic Function eXchange solution.

Syntax

```
pr_recombine [-cell <arg>] [-quiet] [-verbose]
```

Usage

Name	Description
[-cell]	(Required) Specify reconfigurable container module name
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

FileIO

pr_subdivide

Subdivide a Reconfigurable Partition into one or more lower-level Reconfigurable Partitions when using the Hierarchical Dynamic Function eXchange solution.

Syntax

```
pr_subdivide [-cell <arg>] [-subcells <arg>] [-quiet] [-verbose]
    [<from_dcp>]
```

Usage

Name	Description
[-cell]	(Required) Specify parent reconfigurable partition module name
[-subcells]	(Required) Specify child reconfigurable partition module names
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<from_dcp>]	(Required) Specify OOC synthesized checkpoint path for the reconfigurable module specified by option -cell

Categories

FileIO

Send Feedback

pr_verify

Verify whether the design check points are replaceable on board. This command supports these formats: (1) 'pr_verify DCP1 DCP2 -full_check'; (2) 'pr_verify -initial DCP1 -additional {DCP2 DCP3 DCP4 ...}'; (3) 'pr_verify -in_memory -additional {DCP2 DCP3 ...}'; For format (3), it must have an in-memory design opened.

Syntax

```
pr_verify [-full_check] [-file <arg>] [-initial <arg>] [-additional <arg>]
    [-in_memory] [-quiet] [-verbose] [<file1>] [<file2>]
```

Usage

Name	Description
[-full_check]	Default behavior is to report the first difference only; if this option is set to true, pr_verify will report complete difference in placement or routing
[-file]	Filename to output results to. Send output to console if -file is not used.
[-initial]	Initial checkpoint (.dcp)
[-additional]	Additional checkpoints (.dcp)
[-in_memory]	use in-memory design for comparison, combined with -additional option
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<file1>]	Design checkpoint (.dcp) file one
[<file2>]	Design checkpoint (.dcp) file two

Categories

[FileIO](#)

Description

This command is used to compare design checkpoint files for use in the Partial Reconfiguration flow.

For Partial Reconfigurable designs to work in hardware, the placement and routing of static logic must be consistent between all configurations. In addition, proxy logic must be placed in the same locations and clock spine routing must match. The `pr_verify` command compares routed design checkpoint files (DCP) created for a Partial Reconfiguration design to verify that all imported resources match. For more information refer to the *Vivado Design Suite User Guide: Dynamic Function eXchange* (UG909).

The two modes for `pr_verify` let you specify two DCP files to compare, or multiple DCP files to compare against the first DCP file. The syntax for the two modes is:

- pr_verify DCP1 DCP2

- pr_verify -initial DCP1 -additional {DCP2 DCP3 DCP4}

The second mode is the same as repeating the `pr_verify` command to compare each additional DCP with the initial DCP, but keeps the initial DCP open to speed the additional comparisons:

```
pr_verify DCP1 DCP2
pr_verify DCP1 DCP3
pr_verify DCP1 DCP4
```

This command returns the results of the comparison, or returns an error if it fails.

Arguments

`-full_check` - (Optional) Run a complete check of the two design checkpoints to report all mismatched resources. By default the `pr_verify` command will stop after the first mismatch since the two design checkpoints are not valid for partial reconfiguration.

`-file <arg>` - (Optional) Write the comparison results into the specified file. The specified file will be overwritten if one already exists.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-initial <arg>` - (Optional) Specify the initial design checkpoint file to use for comparison.

`-additional` **<args>** - (Optional) Specify one or more additional checkpoints to compare against the `-initial` checkpoint. Multiple checkpoints should be enclosed in quotes or braces.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<file1> - The first design checkpoint file to compare.

<file2> - The second design checkpoint file to compare.

Send Feedback

Examples

The following example compares the two corner DCPs, specified with the `-additional` option, against the inital DCP, running a full check on the designs:

```
pr_verify -full_check -initial FastConfig.dcp \
    -additional {corner1.dcp corner2.dcp}
```

See Also

- read_checkpoint
- write_checkpoint

Send Feedback

program_hw_cfgmem

Program Cfgmem object

Syntax

```
program_hw_cfgmem [-svf_file <arg>] [-force] [-append] [-quiet] [-verbose]
    [<hw_cfgmem>...]
```

Usage

Name	Description
[-svf_file]	svf file to be generated
[-force]	overwrite svf_file if it already exists
[-append]	append to svf file
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<hw_cfgmem>]	list of hardware cfgmems Default: current hardware cfgmem

Categories

Hardware

Description

Erase, blank check, program, and/or verify the specified hw_cfgmem object with the memory configuration file defined in the object's PROGRAM.FILE property. The memory configuration file is created with the `write_cfgmem` command, and associated with the hw_cfgmem object using the `set_property` command as shown in the example.

The process whereby the design specific data is loaded or programmed into the Xilinx FPGA is called configuration. The `create_hw_cfgmem` command defines a flash memory device used for configuring and booting the FPGA device.

After the hw_cfgmem object is created, and associated with the hw_device, the configuration memory can be programmed with the bitstream and other data from a memory configuration file created with the `write_cfgmem` command. The hw_cfgmem object is programmed using the `program_hw_cfgmem` command.

The `program_hw_cfgmem` command will run a multi-step process to erase the configuration memory device, perform a blank check to validate that the device is empty, program the device with the memory configuration file, and verify the programming on the device. Properties on the hw_cfgmem object determine which steps of the programming process are performed. These properties include:

Send Feedback

- PROGRAM.FILES - Specifies the memory configuration files to use for programming the device. The memory configuration files are created with the `write_cfgmem` command.

- PROGRAM.ADDRESS_RANGE - Specifies the address range of the configuration memory device to program. The address range values can be:

  - `{use_file}` - Use only the address space required by the memory configuration file to erase, blank check, program, and verify.

  - `{entire_device}` - Erase, blank check, program, and verify the entire device.

- PROGRAM.ERASE - Erases the contents of the flash memory when true. This is a boolean property with a value of 0 (false) or 1 (true).

- PROGRAM.BLANK_CHECK - Checks the device to make sure the device is void of data prior to programming. This is a boolean property with a value of 0 (false) or 1 (true).

- PROGRAM.CFG_PROGRAM - Program the device with the specified PROGRAM.FILE. This is a boolean property with a value of 0 (false) or 1 (true).

- PROGRAM.VERIFY - Verify the device after programming. This is a boolean property with a value of 0 (false) or 1 (true).

The `program_hw_cfgmem` command can also generate an SVF file for in-system and remote programming of Xilinx devices. SVF is an industry standard file format that is used to describe JTAG chain operations by describing the information that needs to be shifted into the device chain. SVF files are ASCII files that can be written and modified in any text editor. Many third-party programming utilities can use the SVF file to program Xilinx devices in a JTAG chain.

This command returns a transcript of its process when successful, or returns an error if it fails.

Arguments

`-svf_file <arg>` - (Optional) Create an SVF output file for programming SPI and BPI Flash configuration memories. The SVF file can be used by third party tools, and also supports JTAG transaction tracing and improving Bullseye coverage.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-force` - (Optional) Overwrite the specified SVF file if it already exists.

> 💡 **TIP:** *If `-force` is specified with `-append`, the `-append` option is ignored, and a new SVF file is created overwriting the existing file.*

`-append` - (Optional) Append the SVF output to the specified file rather than overwriting it.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Send Feedback

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_cfgmem> - (Required) Specify a hw_cfgmem object to program. The hw_cfgmem must be specified as an object by `get_hw_cfgmems` or `current_hw_cfgmem`, rather than simply by name.

Example

The following example creates a hw_cfgmem object associated with the current_hw_device, defines the memory configuration file created from the bitstream with the `write_cfgmem` command, defines other properties of the programming process, and then programs the hw_cfgmem object:

```
create_hw_cfgmem -hw_device [current_hw_device] [lindex $cfgParts 0 ]
set cfgMem [current_hw_cfgmem]
set_property PROGRAM.FILE {C:/Data/config_n25q128.mcs} $cfgMem
set_property PROGRAM.ADDRESS_RANGE  {use_file} $cfgMem
set_property PROGRAM.BLANK_CHECK  1 $cfgMem
set_property PROGRAM.ERASE  1 $cfgMem
set_property PROGRAM.CFG_PROGRAM  1 $cfgMem
set_property PROGRAM.VERIFY  1 $cfgMem
program_hw_cfgmem $cfgMem
```

Note: In this example, the current hw_cfgmem object is assigned to the variable $cfgMem.

The following example programs the current hw_cfgmem object:

```
program_hw_cfgmem [current_hw_cfgmem]
```

See Also

- create_hw_cfgmem
- current_hw_cfgmem
- current_hw_device
- delete_hw_cfgmem
- get_cfgmem_parts
- get_hw_cfgmems
- get_property
- readback_hw_cfgmem
- set_property
- write_cfgmem

Send Feedback

program_hw_devices

Program hardware devices

Syntax

```
program_hw_devices [-key <arg>] [-clear] [-skip_program_keys]
    [-skip_program_rsa] [-user_efuse <arg>] [-user_efuse_128 <arg>]
    [-control_efuse <arg>] [-security_efuse <arg>] [-only_export_efuse]
    [-svf_file <arg>] [-efuse_export_file <arg>] [-disable_eos_check]
    [-skip_reset] [-force] [-append] [-type <arg>] [-quiet] [-verbose]
    [<hw_device>...]
```

Returns

Hardware devices

Usage

Name	Description
[-key]	key option value for encryption programming: efuse,bbr,none
[-clear]	clear bbr registers, only valid for bbr
[-skip_program_keys]	skip programming keys specified in NKY file, if any
[-skip_program_rsa]	skip programming RSA key specified in NKY file, if any
[-user_efuse]	hex user fuse value for encryption programming
[-user_efuse_128]	hex user fuse 128 bit value for encryption programming
[-control_efuse]	hex control fuse value for encryption programming
[-security_efuse]	hex security fuse value for encryption programming
[-only_export_efuse]	do not program eFUSE; just export settings to efuse_export_file
[-svf_file]	svf file used to program device
[-efuse_export_file]	output file to store programmed eFUSE settings
[-disable_eos_check]	Disables End of Startup check after programming
[-skip_reset]	Skips resetting the device before programming
[-force]	overwrites svf file and creates empty file
[-append]	append to svf file
[-type]	bitstream file type to be used for programming: bit,bin,rbt
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<hw_device>]	list of hardware devices Default: current hardware device

Categories

Hardware

Description

Program the specified hardware device object or objects on the open hardware target of the current hardware server.

To access a Xilinx FPGA through the Hardware Manager, you must use the following Tcl command sequence:

1. `open_hw` - Opens the Hardware Manager in the Vivado Design Suite.
2. `connect_hw_server` - Makes a connection to a local or remote Vivado hardware server application.
3. `current_hw_target` - Defines the hardware target of the connected server.
4. `open_hw_target` - Opens a connection to the hardware target.
5. `current_hw_device` - Specifies the Xilinx FPGA to use for programming and debugging.

After connecting to the target hardware device, you must associate the bitstream file (`.bit`, `.rbt`, `.bin`) from the design with the device, using the set_property command:

```
set_property PROGRAM.FILE {C:/Data/design.bit} [current_hw_device]
```

For debug purposes, you can also associate a probes file (`.ltx`) with the device using the PROBES.FILE property:

```
set_property PROBES.FILE {C:/Data/debug_nets.ltx} [current_hw_device]
```

Once the programming file has been associated with the hardware device, you can program the hardware device using the `program_hw_devices` command, and debug the device using any of a number of Hardware Manager Tcl commands. To interactively debug the device open the Hardware Manager in the Vivado Design Suite IDE.

You can also program an encrypted bitstream into the specified hw_device. This requires the implemented design to have encryption properties assigned prior to generating the bitstream with the `write_bitstream` command. You can add ENCRYPTION properties to the design most easily using the Encryption page of the Edit Device Properties dialog box in the Vivado IDE. Refer to the *Vivado Design Suite User Guide: Programming and Debugging (UG908)* for more information on the Edit Device Properties dialog box.

Programming a device for an encrypted bitstream is a two-step process that requires running `program_hw_devices` once to program the encryption key into the BBR or eFUSE registers, and a second time to program the encrypted bitstream into the device:

```
program_hw_devices -key bbr [current_hw_device]
program_hw_device [current_hw_device]
```

> ⚠️ **CAUTION!** *eFUSEs are one-time programmable cells on the hardware device, used to store the factory-programmed Device DNA, AES-GCM encryption key, and user specified values. Refer to the UltraScale Architecture Configuration (UG570) or 7 Series FPGAs Configuration User Guide (UG470) for more information on eFUSE registers.*

The `program_hw_devices` command can also generate a Serial Vector Format (SVF) file for in-system and remote programming of Xilinx devices. SVF is an industry standard file format that is used to describe JTAG chain operations by describing the information that needs to be shifted into the device chain. SVF files are ASCII files that can be written and modified in any text editor. Many third-party programming utilities can use the SVF file to program Xilinx devices in a JTAG chain.

This command returns a transcript of its actions, or returns an error if it fails.

Arguments

`-key [ bbr | efuse ]` - (Optional) Specify that the encryption key should be programmed into the battery-backed SRAM or the eFUSE registers of the specified hw device. The encryption key is defined in the encryption key (NKY or NKZ) file generated by the `write_bitstream` command for encrypted bit files, and is associated with the hw_bitstream object with the ENCRYPTION.FILE property. The key value is extracted from the file and is stored in the ENCRYPTION.KEY property of the hw_bitstream object associated with the hardware device.

> 💡 **TIP:** *The encryption properties of the bitstream must be defined prior to creating the bitstream with the* `write_bitstream` *command.*

`-clear` - (Optional) Clear the encryption key from the currently programmed battery-backed SRAM (BBR). Use the `-clear` option with `-key` and with the hw_device to clear the encryption key from the BBR.

`-user_efuse <arg>` - (Optional) Specify a 32-bit value to program into the FUSE_USER eFUSE Register of the hw_device or devices.

`-user_efuse_128 <arg>` - (Optional) Specify a 128-bit value to program into the FUSE_USER eFUSE Register of the hw_device or devices.

`-control_efuse <arg>` - (Optional) Specify a value to program into the FUSE_CNTL eFUSE Register of the hw_device or devices. The specified value can be a 14 bit number for 7 series devices, or a 21 bit number for UltraScale devices.

`-security_efuse <arg>` - (Optional) Specify a 256-bit value to program into the security eFUSE Register of the hw_device or devices.

`-only_export_efuse` - (Optional) Do not program the eFUSE registers; just export the eFUSE settings to the file specified by `-efuse_export_file`. This lets you check the eFUSE programming for the device before committing it to the eFUSE registers.

`-svf_file <arg>` - (Optional) Create an SVF output file while programming the device. The SVF file can be used by third party tools, and also supports JTAG transaction tracing and improving Bullseye coverage. You can also generate an SVF file from a programmed hw_device using the `write_hw_svf` command.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-efuse_export_file <arg>` - (Optional) Specifies an output file (`.nkz`) to write the current eFUSE settings. If this option is not specified, the current eFUSE settings will be written to a file named `export_<FUSE_DNA>.nkz`, where <FUSE_DNA> is the DNA value from the device.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-disable_eos_check` - (Optional) Disables "End of Startup" (EOS) check after programming.

`-force` - (Optional) Overwrite the specified SVF file or eFUSE file if one already exists.

> **TIP:** *If* `-force` *is specified with* `-append`*, the* `-append` *option is ignored, and a new SVF file is created overwriting the existing file.*

`-append` - (Optional) Append the SVF output to the specified file rather than overwriting it.

`-type [ bit | bin | rbt ]` - (Optional) Specifies the type of bitstream file as either a standard bitstream (`.bit`), a binary bitstream without header (`.bin`), or a raw bitstream which is an ASCII file (`.rbt`).

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Displays the eFUSE register values of the programmed hw_device or devices.

<hw_device> - (Optional) One or more hw_device objects to program. The hw_device must be specified as an object as returned by the `get_hw_devices` command. If the device is not specified, the `current_hw_device` will be programmed.

Example

The following example sets the current hw_device object, then sets the PROGRAM.FILE property for the device, and programs the device:

```
current_hw_device [lindex [get_hw_devices] 0]
set_property PROGRAM.FILE {C:/Data/design.bit} [current_hw_device]
program_hw_devices [current_hw_device]
```

The following example programs the encryption key into the battery-backed RAM (BBR), and then programs the current hw_device:

```
program_hw_devices -key bbr [current_hw_device]
program_hw_devices [current_hw_device]
```

The following example writes an SVF file from the programmed device:

```
program_hw_devices -force -svf_file {C:/Data/test1.svf} [current_hw_device]
```

This example clears the BBR encryption key programming:

```
program_hw_devices -key bbr -clear [current_hw_device]
```

See Also

- connect_hw_server
- create_hw_device
- create_hw_target
- current_hw_device
- current_hw_target
- get_hw_devices
- get_hw_targets
- open_hw_target
- verify_hw_devices
- write_bitstream
- write_hw_svf

Send Feedback

ptrace

Turns on or off printing of name of the hdl process being simulated

Syntax

```
ptrace [-quiet] [-verbose] <value>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<value>	value: on, true, yes. Otherwise set to off, false, no

Categories

Simulation

Description

Enables process tracing for simulation debugging purposes.

During simulation the name of the HDL process that is evaluated will be written to the Tcl console, as well as the simulation source file and line number associated with the process.

> **TIP:** *Process tracing provides more detailed information than is available with line tracing and the* `ltrace` *command.*

This feature can also be enabled using the PROCESS_TRACING property on the current simulation object:

```
set_property PROCESS_TRACING on [current_sim]
```

The command returns the state of process tracing, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

Send Feedback

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<value>` - (Required) Enables or disables process tracing during simulation. Specify a `<value>` of true to enable process tracing, or false to disable it.

Example

The following example enables process tracing:

```
ptrace true
```

See Also

- current_sim
- ltrace
- set_property

read_bd

Read one or more IPIntegrator design files

Syntax

```
read_bd [-quiet] [-verbose] <files>...
```

Returns

List of IPIntegrator design file objects that were added.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<files>	IPIntegrator design file name(s)

Categories

FileIO, IPIntegrator

Description

Read the specified IP subsystem design files, or block designs, into the current project or the in-memory design. This command is similar to the `add_files` command. The block design file is added to the source fileset as it is read.

Note: Files are read and referenced from their current location, and are not moved into the local project directories. To bring the file into the local project, use the `import_files` command instead.

You can use this command to read block designs into the in-memory design, when running the Vivado tool in Non Project mode, in which there is no project file to maintain and manage the various project source files. Refer to the *Vivado Design Suite User Guide: Design Flows Overview* (UG892) for more information on Non Project mode.

This command returns the name of the IP subsystem design files read, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Send Feedback

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<files>` - Specify the name of the IP subsystem files to read into the current project or in-memory design.

Note: If the path is not specified as part of the file name, the tool will search for the specified file in the current working directory and then in the directory from which the tool was launched.

Examples

The following example reads the specified IP subsystem design into the current project:

```
read_bd C:/Data/block_designs/design1.bd
```

See Also

- add_files
- import_files
- open_bd_design
- save_bd_design

Send Feedback

read_checkpoint

Read a design checkpoint

Syntax

```
read_checkpoint [-cell <arg>] [-incremental] [-directive <arg>]
    [-auto_incremental] [-reuse_objects <args>] [-fix_objects <args>]
    [-dcp_cell_list <args>] [-quiet] [-verbose] [<file>]
```

Usage

Name	Description
[-cell]	Replace this cell with the checkpoint. The cell must be a black box.
[-incremental]	Input design checkpoint file to be used for re-using implementation.
[-directive]	Mode of behavior (directive) for this command. Please refer to Arguments section of this help for values for this option. Default: RuntimeOptimized
[-auto_incremental]	Enters automatic mode for incremental compile. This is less aggressive than the standard incremental mode and will readily switch from incremental algorithms to the default algorithms if results might not be maintained. Should not be used with -reuse_objects.
[-reuse_objects]	Reuse only given list of cells, clock regions, SLRs and Designs
[-fix_objects]	Fix only given list of cells, clock regions, SLRs or Design
[-dcp_cell_list]	A list of cell/dcp pairs, e.g. {<cell1> <dcp1> <cell2> <dcp2>}. The option value should be in curly braces.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<file>]	Design checkpoint file

Categories

[FileIO](#)

Description

Reads a design checkpoint file (DCP) that contains the netlist, constraints, and may optionally have the placement and routing information of an implemented design. You can save design checkpoints at any stage in the design using the `write_checkpoint` command.

Send Feedback

The `read_checkpoint` command simply reads the associated checkpoint file, without opening a design or project in-memory. To create a project from the imported checkpoint, use the `open_checkpoint` command instead of `read_checkpoint`, or use the `link_design` command after `read_checkpoint` to open the in-memory design from the checkpoint or checkpoint files currently read.

Note: When multiple design checkpoints are open in the Vivado tool, you must use the `current_project` command to switch between the open designs. You can use `current_design` to check which checkpoint is the active design.

> **IMPORTANT!** *The `-incremental` switch is not intended to merge two DCP files into a single design. It applies the placement and routing of the incremental checkpoint to the netlist objects in the current design.*

After loading an incremental design checkpoint, you can use the `report_incremental_reuse` command to determine the percentage of physical data reused from the incremental checkpoint, in the current design. The `place_design` and `route_design` commands will run incremental place and route, preserving reused placement and routing information and incorporating it into the design solution.

Reading a design checkpoint with `-incremental`, loads the physical data into the current in-memory design. To clear out the incremental design data, you must either reload the current design, using `open_run` to open the synthesis run for instance, or read a new incremental checkpoint to overwrite the one previously loaded.

Arguments

`-cell <arg>` - (Optional) Specifies a black box cell in the current design to populate with the netlist data from the checkpoint file being read. This option cannot be used with `-incremental`, or any of its related options.

`-auto_incremental` - (Optional) Enables incremental algorithms only when the quality of the reference checkpoint is high. This allows a user to set this option and let the tool decide whether to run default or incremental algorithms.

`-incremental` - (Optional) Load a checkpoint file into an already open design to enable the incremental implementation design flow, where `<file>` specifies the path and filename of the incremental design checkpoint (DCP) file. In the incremental implementation flow, the placement and routing from the incremental DCP is applied to matching netlist objects in the current design to reuse existing placement and routing. Refer to the *Vivado Design Suite User Guide: Implementation (UG904)* for more information on incremental implementation.

`-directive [ RuntimeOptimized | TimingClosure | Quick ]` - (Optional) Specifies a directive, or mode of operation for the `-incremental` process.

- RuntimeOptimized: The target WNS is the referenced from the incremental DCP. By default, more reuse from the reference checkpoint is encouraged.

- TimingClosure: The target WNS is 0. This mode attempts to meet timing at the expense of runtime. Paths not meeting timing will be ripped up and the incremental algorithms will be applied to fix timing on the paths.

- Quick: Specifies a low effort, non-timing-driven incremental implementation mode with the fastest runtime.

`-reuse_objects` <args> - (Optional) For use with the `-incremental` option, to read and reuse only a portion of the checkpoint file, this option specifies to reuse only the placement and routing data of the specified list of cells, clock regions, and SLRs from the incremental checkpoint.

> 💡 **TIP:** *When this option is not specified, the whole design will be reused. The* `-reuse_objects` *options can be used multiple times to reuse different object types. See examples below.*

`-fix_objects` - (Optional) When `-incremental` is specified, mark the placement location of specifed cells as fixed (IS_LOC_FIXED) to prevent changes by the `place_design` command. This option will fix the placement of the specified list of cells, clock regions, SLRs or the `current_design`.

`-dcp_cell_list` <arg> - (Optional) Lets you specify a list of cell/dcp pairs. This lets you read in multiple DCP files for specified cells in a single run of the `read_checkpoint` command. The format is specified as a list of cell/DCP pairs enclosed in curly braces: `{<cell1> <dcp1> <cell2> <dcp2>}`.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file>` - (Required) The path and filename of the checkpoint file to read.

Note: If the path is not specified as part of the file name, the tool will search for the specified file in the current working directory and then in the directory from which the tool was launched.

Examples

The following example imports the specified checkpoint file into the tool, and then links the various design elements to create an in-memory design of the specified name:

```
read_checkpoint C:/Data/checkpoint.dcp
link_design -name Test1
```

This example reads a design checkpoint on top of the current design for incremental place and route of the design:

```
read_checkpoint -incremental C:/Data/routed.dcp
```

This example reads a design checkpoint. Incremental algorithms will be used only if the timing is nearly met and if there is a high reuse:

```
read_checkpoint -auto_incremental C:/Data/routed.dcp
```

Reuse and fix the placement and routing associated with the DSPs and Block RAMs:

```
read_checkpoint -incremental C:/Data/routed.dcp \
-reuse_objects [all_rams] -reuse_objects [all_dsps] -fix_objects
[current_design]
```

> 💡 **TIP:** *The* `-reuse_objects` *option could also be written as:*
>
> ```
> -reuse_objects [get_cells -hier -filter {PRIMITIVE_TYPE =~ BMEM.*.* ||
> PRIMITIVE_TYPE =~ MULT.dsp.* }]
> ```

The following example reuses the placement and routing of the cells inside the hierarchical cpuEngine cell, and fixes the placement of the DSP cells:

```
read_checkpoint -incremental C:/Data/routed.dcp -reuse_objects [get_cells
cpuEngine] -fix_objects [all_dsps]
```

See Also

- all_dsps
- config_implementation
- current_design
- current_project
- get_cells
- link_design
- open_checkpoint
- report_config_implementation
- write_checkpoint

Send Feedback

read_csv

Import package pin and port placement information

Syntax

```
read_csv [-quiet_diff_pairs] [-quiet] [-verbose] <file>
```

Usage

Name	Description
[-quiet_diff_pairs]	Suppress warnings about differential pair inference when importing I/O ports
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<file>	Pin Planning CSV file

Categories

FileIO

Description

Imports port definition and package pin placement information from a comma separated value (CSV) file.

The port definitions in a CSV file can be imported into an I/O Pin Planning project. In a Pin Planning project, importing a CSV file replaces the current port definitions. Any ports in the design that are not found in the imported CSV file will be removed.

In all other projects the port definitions are defined in the source design data, however package pin assignments and port attributes can be read from the specified CSV file.

The ports read from the CSV file can not have spaces in the name, or the tool will return an error. The specific format and requirements of the CSV file are described in the *Vivado Design Suite User Guide: I/O and Clock Planning (UG899)*.

Arguments

-quiet_diff_pairs - (Optional) The tool transcripts messages related to pins that may be inferred as differential pairs when importing the CSV file. This option suppresses messages related to inferring differential pairs.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Send Feedback

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file>` - (Required) The file name of the CSV file to be imported.

Note: If the path is not specified as part of the file name, the tool will search for the specified file in the current working directory and then in the directory from which the tool was launched.

Examples

The following example imports a CSV file into an open project:

```
read_csv C/Data/pinList.csv
```

The following example sets up a new IO Pin Planning project, and then imports the specified CSV file into it, and infers any differential pairs in the CSV file:

```
create_project myPinPlan C:/Data/myPinPlan -part xc7v285tffg1157-1
set_property design_mode PinPlanning [current_fileset]
open_io_design -name io_1
read_csv C:/Data/import.csv
infer_diff_pairs -filetype csv C:/Data/import.csv
```

Note: The design_mode property on the source fileset is what determines the nature of the project.

See Also

- create_project
- infer_diff_pairs
- open_io_design
- set_property
- write_csv

read_edif

Read one or more EDIF or NGC files

Syntax

```
read_edif [-quiet] [-verbose] <files>
```

Returns

List of file objects that were added.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<files>	EDIF or NGC file name(s)

Categories

FileIO

Description

Imports an EDIF or NGC netlist file into the Design Source fileset of the current project.

> ⭐ **IMPORTANT!** *NGC format files are not supported in the Vivado Design Suite for UltraScale devices. It is recommended that you regenerate the IP using the Vivado Design Suite IP customization tools with native output products. Alternatively, you can use the* `NGC2EDIF` *command to migrate the NGC file to EDIF format for importing. For more information refer to the ISE to Vivado Design Suite Migration Guide (UG911).*

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<files>` - (Required) The name of the EDIF or NGC files to be imported.

Note: If the path is not specified as part of the file name, the tool will search for the specified file in the current working directory and then in the directory from which the tool was launched.

Examples

The following example imports an EDIF file into the open project:

```
read_edif C/Data/bft_top.edf
```

See Also

- write_edif

read_hw_ila_data

Read hardware ILA data from a file

Syntax

```
read_hw_ila_data [-quiet] [-verbose] <file>
```

Returns

Name of the output file.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<file>	hardware ILA data file name

Categories

[Hardware](#)

Description

Read ILA debug core data from the specified file, and create an hw_ila_data object.

The ILA debug sample data is acquired from a running device using the `upload_hw_ila_data` command. This creates a hw_ila_data object that can be written to a file on disk using the `write_hw_ila_data` command. This command reads that ILA data file.

The hw_ila_data object that is created by `read_hw_ila_data` is named after the `<file>` it is read from. If a `hw_ila_data` object of the same name already exists, the name of the object is assigned a number extension starting at 1: `<file>_1`.

The new hw_ila_data object is not connected with, or associated with, any ILA debug cores in the design.

After being read from disk, the ILA debug data can be viewed in the waveform viewer of the Vivado logic analyzer by using the `display_hw_ila_data` command.

This command returns an ILA data object, or returns an error if it fails.

Send Feedback

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file>` - (Required) The name of ILA data file to read. If the file extension is not specified, the Vivado tool assumes an extension of `.ila`.

Note: If the path is not specified as part of the file name, the tool will search for the specified file in the current working directory and then in the directory from which the tool was launched.

Example

The following example reads the specified ILA data file, and creates a hw_ila_data object named after the file:

```
read_hw_ila_data C:/Data/hw_ila_data_2.ila
```

See Also

- current_hw_ila
- current_hw_ila_data
- display_hw_ila_data
- get_hw_ilas
- get_hw_ila_datas
- run_hw_ila
- write_hw_ila_data

read_hw_sio_scan

Read hardware SIO scan data from a file. A hardware SIO scan object will be created if not provided.

Syntax

```
read_hw_sio_scan [-quiet] [-verbose] <file> [<hw_sio_scan>]
```

Returns

Hardware SIO scan object.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<file>	hardware SIO scan file name
[<hw_sio_scan>]	hardware SIO scan data object Default: None

Categories

[Hardware](#)

Description

Read a hardware SIO scan data file and create a hw_sio_scan object in the Hardware Manager feature of the Vivado Design Suite.

The SIO scan data can be written to disk using the `write_hw_sio_scan` command, after running the scan using the `run_hw_sio_scan` command. This command reads that data file.

If no hw_sio_scan object is specified, a new hw_sio_scan object is created and is named sequentially following any existing hw_sio_scan objects. After being read from disk, the SIO scan data can be plotted and viewed in the Vivado serial I/O analyzer by using the `display_hw_sio_scan` command.

This command returns a hw_sio_scan object, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file>` - (Required) The name of SIO scan data file to read. If the file extension is not specified, the Vivado tool assumes an extension of `.csv`.

Note: If the path is not specified as part of the file name, the tool will search for the specified file in the current working directory and then in the directory from which the tool was launched.

<hw_sio_scan> - (Optional) Specify an existing hw_sio_scan object to read the scan data file into. The hw_sio_scan object can be specified by name, or as an object returned by the `get_hw_sio_scans` command.

Example

The following example reads the specified SIO scan data file into an existing hw_sio_scan object:

```
read_hw_sio_scan C:/Data/LoopBack1.csv SCAN_0
```

See Also

- create_hw_sio_scan
- current_hw_device
- display_hw_sio_scan
- get_hw_sio_scans
- remove_hw_sio_scan
- run_hw_sio_scan
- stop_hw_sio_scan
- wait_on_hw_sio_scan
- write_hw_sio_scan

read_hw_sio_sweep

Read hardware SIO sweep data from a directory. A hardware SIO sweep object will be created if not provided.

Syntax

```
read_hw_sio_sweep [-quiet] [-verbose] <directory> [<hw_sio_sweep>]
```

Returns

Hardware SIO sweep object.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<directory>	hardware SIO sweep directory name
[<hw_sio_sweep>]	hardware SIO sweep data object Default: None

Categories

Hardware

Description

Read a hardware SIO sweep data directory and create a hw_sio_sweep object in the Hardware Manager feature of the Vivado Design Suite.

The SIO sweep data can be written to disk using the `write_hw_sio_sweep` command, after running the sweep using the `run_hw_sio_sweep` command. This command reads the sweep directory containing multiple SIO scan data files.

If no hw_sio_sweep object is specified, a new hw_sio_sweep object is created and is named sequentially following any existing hw_sio_sweep objects. After being read from disk, any of the SIO scans in the sweep can be plotted and viewed in the Vivado serial I/O analyzer by using the `display_hw_sio_scan` command.

This command returns a hw_sio_sweep object, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<directory> - (Required) The name of the dirctory containing the SIO sweep data to read.

<hw_sio_sweep> - (Optional) Specify an existing hw_sio_sweep object to read the scan data files of the sweep into. The hw_sio_sweep object can be specified by name, or as an object returned by the `get_hw_sio_sweeps` command.

Example

The following example reads the specified sweep directory and creates a new hw_sio_sweep object, then displays one of the hw_sio_scans from that sweep:

```
read_hw_sio_sweep C:/Data/SWEEP_1/
display_hw_sio_scan [get_hw_sio_scans {SCAN_86}]
```

See Also

- create_hw_sio_sweep
- current_hw_device
- display_hw_sio_scan
- get_hw_sio_sweeps
- remove_hw_sio_sweep
- run_hw_sio_sweep
- stop_hw_sio_sweep
- wait_on_hw_sio_sweep
- write_hw_sio_sweep

read_ip

Read one or more IP files

Syntax

```
read_ip [-quiet] [-verbose] <files>
```

Returns

List of IP file objects that were added.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<files>	IP file name(s)

Categories

[FileIO](), [IPFlow]()

Description

Read the specified list of IP files (XCI) and add them to the design and the current fileset. Files are added by reference into the current project, just as in the `add_files` command.

You can use this command to read the contents of source files into the in-memory design, when running the Vivado tool in Non Project mode, in which there is no project file to maintain and manage the various project source files. Refer to the *Vivado Design Suite User Guide: Design Flows Overview* (UG892) for more information on Non Project mode.

When using the `read_ip` command all output products associated with the IP core, including the design checkpoint file (DCP) will be read into the in-memory design.

> 💡 **TIP:** *In the project-based design flow, the Vivado tool will automatically generate the necessary output products associated with an IP core. However, in a non-project flow you must generate the necessary output products using the* `synth_ip` *or* `generate_target` *commands. For more information on working with IP refer to the Vivado Design Suite User Guide: Designing with IP (UG896).*

Use the `import_ip` command to add the IP cores and import the files into the local project directory.

This command returns the list of files read.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<files>` - (Required) The list of IP files to read into the current project. Both XCI and XCO file formats are supported. An XCI file is an IP-XACT format file that contains information about the IP parameterization. An XCO file is a CORE Generator log file that records all the customization parameters used to create the IP core and the project options in effect when the core was generated.

Examples

The following example reads the specified IP files:

```
read_ip C:/test_ip/char_fifo.xci
```

See Also

- add_files
- import_ip

read_iphys_opt_tcl

load iPhysOpt script and run it.

Syntax

```
read_iphys_opt_tcl [-fanout_opt] [-critical_cell_opt] [-replicate_cell]
    [-placement_opt] [-rewire] [-forward_retime] [-backward_retime]
    [-dsp_register_opt] [-bram_register_opt] [-uram_register_opt]
    [-shift_register_opt] [-shift_register_to_pipeline] [-auto_pipeline]
    [-pipeline_to_shift_register] [-critical_pin_opt] [-restruct_opt]
    [-equ_drivers_opt] [-include_skipped_optimizations] [-create_bufg]
    [-insert_negative_edge_ffs] [-hold_fix] [-slr_crossing_opt] [-quiet]
    [-verbose] [<input>]
```

Usage

Name	Description
[-fanout_opt]	Fanout optimization including very high fanout optimizations
[-critical_cell_opt]	Do cell-duplication based optimization on timing critical nets
[-replicate_cell]	Atomic cell replication operation for interconnect retiming and LUT optimization
[-placement_opt]	Move cells to reduce delay on timing-critical nets
[-rewire]	Do rewiring optimization
[-forward_retime]	Forward retiming optimization
[-backward_retime]	Backward retiming optimization
[-dsp_register_opt]	DSP register optimization
[-bram_register_opt]	BRAM register optimization
[-uram_register_opt]	UltraRAM register optimization
[-shift_register_opt]	Shift register optimization
[-shift_register_to_pipeline]	Shift register to pipeline optimization
[-auto_pipeline]	Auto pipeline
[-pipeline_to_shift_register]	pipeline to shift register optimization
[-critical_pin_opt]	Pin Swap optimization
[-restruct_opt]	LUT restructuring optimization
[-equ_drivers_opt]	Equivalent driver rewiring
[-include_skipped_optimizations]	Apply undo changes
[-create_bufg]	Insert bufg to drive HFO nets
[-insert_negative_edge_ffs]	Inserting negative edge triggered FFs for high hold mitigation
[-hold_fix]	Inserting buffers for hold fix optimization
[-slr_crossing_opt]	Optimize slr crossing nets
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<input>]	iPhysOpt.tcl file

Categories

Tools

Description

Interactive physical optimization can be used in two ways:

- Applying post-placement physical optimizations to the pre-placement netlist to improve the overall placement result and improve design performance.

- Saving the physical optimizations in a Tcl script to be repeated as needed.

To apply post-placement optimizations to the pre-placement netlist, you can reset the implementation run and open the synthesized design, or open the opt_design checkpoint, and read the iphys_opt Tcl script to apply the physical optimizations.

You can apply all optimizations from the iphys_opt Tcl script, or apply specific optimizations using the options of the `read_iphys_opt_tcl` command. You can also include any optimizations that were defined but skipped during physical optimization.

If the iphys_opt Tcl script includes placement data, you can use that data to place the optimized cells in the design.

After reading the iphys_opt Tcl script, and placing the optimized cells, you can rerun placement for the overall design. The design now incorporates the benefits of the `phys_opt_design` optimizations before placement, such as fewer high-fanout nets due to replication, and fewer long distance paths from block RAM outputs. The results should be a better placement, and improved design performance, due to the early application of netlist optimizations.

This command returns a transcript of its processes, or returns an error if it fails.

Arguments

`-fanout_opt` - (Optional) Apply the fanout optimizations that are defined in the specified interactive physical optimization Tcl script.

`-critical_cell_opt` - (Optional) Applies the cell replication optimizations that are defined in the specified Tcl script.

`-placement_opt` - (Optional) Applies the cell placement optimizations that are defined in the specified Tcl script.

`-rewire` - (Optional) Applies the logic cone refactoring that are defined in the specified Tcl script.

`-dsp_register_opt` - (Optional) Applies the DSP optimizations that are defined in the specified interactive physical optimization Tcl script.

`-bram_register_opt` - (Optional) Applies the BRAM optimizations that are defined in the specified Tcl script.

`-shift_register_opt` - (Optional) Applies the shift register optimization that are defined in the specified Tcl script.

`-critical_pin_opt` - Applies the pin-swapping that are defined in the specified Tcl script.

`-include_skipped_optimizations` - (Optional) Apply the skipped optimizations that are defined in the input Tcl script, as well as the standard optimizations. These are optimizations identified by phys_opt_design that are skipped because suitable locations for optimized logic cannot be found. When this option is specified, the `iphys_opt_design` command will attempt to use the included skipped optimizations in the pre-placement netlist.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<input> - (Required) The name of the interactive physical optimization Tcl file to read.

Note: If the path is not specified as part of the file name, the tool will search for the specified file in the current working directory and then in the directory from which the tool was launched.

Examples

The following example applies the BRAM optimizations that are defined in the specified interactive physical optimization Tcl script, and applies any placement data for the optimized cells:

```
open_checkpoint C:/Data/opt_design.dcp
read_iphys_opt_tcl -shift_register_opt -placement_opt C:/Data/
my_iphys_opt.tcl
```

See Also

- iphys_opt_design
- phys_opt_design
- report_phys_opt
- write_iphys_opt_tcl

read_mem

Read one or more data files (.mem .mif .dat).

Syntax

```
read_mem [-quiet] [-verbose] <files>...
```

Returns

List of file objects that were added.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<files>	Data (.mem .coe .dat) file name(s)

Categories

FileIO

Description

This command reads memory files of type MEM, DAT, or COE, and adds the files to the in-memory design, or the current project, to initialize BRAM memory for behavioral simulation, synthesis and post-synthesis simulation.

If the memory is not initialized in the design, then it will be initialized to all 0s.

This command returns the name of the files read, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Send Feedback

`<files>` - (Required) The name of the MEM, DAT, or COE file to read.

Note: If the path is not specified as part of the file name, the tool will search for the specified file in the current working directory and then in the directory from which the tool was launched.

Examples

The following example:

```
read_mem  C:/Data/design1.mem
```

See Also

- generate_mem_files
- write_bmm

read_qor_suggestions

Read QoR Suggestions from the given file

Syntax

```
read_qor_suggestions [-quiet] [-verbose] <file>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<file>	QoR suggestions file Values: Path to the QoR suggestions file, typically ending with .rqs.

Categories

FileIO, Feasibility, Timing

Description

Read the specified file and extract the QoR suggestions from the file.

This command returns the name of the QoR file read, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file>` - (Required) Read the QoR suggestions from the specified file, which should have the `.rqs` suffix.

Note: If the path is not specified as part of the file name, the tool will search for the specified file in the current working directory and then in the directory from which the tool was launched.

Examples

The following example reads the QoR suggestions from the specified file, and then reports the QoR suggestions in the open design:

```
read_qor_suggestions C:/Data/qor_results.rqs
report_qor_suggestions -of_objects [get_qor_suggestions]
```

See Also

- get_qor_suggestions
- report_qor_suggestions
- write_qor_suggestions

Send Feedback

read_saif

Import simulation data in saif format

Syntax

```
read_saif [-strip_path <arg>] [-no_strip] [-out_file <arg>] [-quiet]
    [-verbose] <file>
```

Usage

Name	Description
[-strip_path]	Specifies the name of the instance of the current design as it appears in the SAIF file
[-no_strip]	Do not strip first two levels of hierarchy from SAIF file
[-out_file]	Specifies the name of the output file that contains nets that could not be matched
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<file>	Specifies the name of the SAIF file to be read

Categories

FileIO, Power, Simulation

Description

Reads a Switching Activity Interchange Format (SAIF) file for use during power analysis by the `report_power` command, or power optimization by `power_opt_design`. The `read_saif` command will annotate the design nodes with activity from the SAIF file and estimate power appropriately.

Running `report_power` or `power_opt_design` after reading the SAIF file will use the activity rates from the specified file during optimization and analysis.

Arguments

`-strip_path <arg>` - (Optional) Strip the specified instance path prefix from elements in the SAIF file to allow them to be mapped properly to instances in the current design.

> 💡 **TIP:** *The instance path specified should not begin with a '/'. The `read_saif` parser looks for design net names, which do not have a leading '/'.*

`-no_strip` - (Optional) Do not strip first two levels of hierarchy from the SAIF file.

`-out_file <arg>` - (Optional) The name of an output file where unmatched nets and other messages are reported. This file is created during the import of the SAIF file. If the `-out_file` option is not specified, the information is not saved to a file.

Note: If the path is not specified as part of the file name, the tool will write the specified file to the current working directory, or the directory from which the tool was launched.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file>` - (Required) The name of the SAIF file to read.

Note: If the path is not specified as part of the file name, the tool will search for the specified file in the current working directory and then in the directory from which the tool was launched.

Examples

The following example:

```
read_saif -strip_path design/top/F1 C:/Data/design1.saif
```

See Also

- power_opt_design
- report_power

read_schematic

Import schematic

Syntax

```
read_schematic [-name <arg>] [-quiet] [-verbose] <file>
```

Returns

Name of the file previously exported.

Usage

Name	Description
[-name]	Schematic window title
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<file>	Input file

Categories

FileIO

Description

Import a native schematic file that was previously exported from the Vivado Design Suite using the `write_schematic` command.

Arguments

`-name <arg>` - (Optional) The name of the Schematic window to open when reading the schematic file.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<files>` - (Required) The name of the Schematic file to read.

Note: If the path is not specified as part of the file name, the tool will search for the specified file in the current working directory and then in the directory from which the tool was launched.

Example

The following example reads the specified schematic file and opens a Schematic window called "Sheet_1" in the Vivado IDE:

```
read_schematic C:/Data/mySchematic.txt -name Sheet_1
```

See Also

- write_schematic

read_twx

Read timing results from Trace STA tool

Syntax

```
read_twx [-cell <arg>] [-pblock <arg>] [-quiet] [-verbose] <name> <file>
```

Usage

Name	Description
[-cell]	Interpret names in the report file as relative to the specified cell
[-pblock]	Interpret names in the report file as relative to the specified pblock
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<name>	Name for the set of results
<file>	Name of the Trace import file

Categories

FileIO

Description

Imports timing results in the TWX format timing report files generated by the Xilinx Timing Reporter And Circuit Evaluator (TRACE) tool. The TWX file can be imported at the top-level, which is the default, or at a specific cell-level or relative to a specific Pblock.

After the TWX files are imported, the timing results display in the Timing Results view in GUI mode.

Arguments

-cell <arg> - (Optional) Specify The name of a hierarchical cell in the current design to import the TWX file into. The timing paths will be applied to the specified cell.

-pblock <arg> - (Optional) The name of a Pblock in the current design. The timing paths will be imported relative to the specified block.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - (Required) The name of the Timing Results view to create when importing the timing paths in the TWX file.

Note: Both `<name>` and `<file>` are required positional arguments. The `<name>` argument must be provided first.

`<file>` - (Required) The file name of the TWX file to be imported.

Note: If the path is not specified as part of the file name, the tool will search for the specified file in the current working directory and then in the directory from which the tool was launched.

Examples

The following example reads the specified TWX file into the top-level of the design:

```
read_twx C:/Data/timing_files/bft.twx
```

See Also

- report_timing

Send Feedback

read_verilog

Read one or more Verilog files

Syntax

```
read_verilog [-library <arg>] [-sv] [-quiet] [-verbose] <files>...
```

Returns

List of file objects that were added.

Usage

Name	Description
[-library]	Library name (ignored by Vivado synthesis) Default: default lib
[-sv]	Enable system verilog compilation
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<files>	Verilog file name(s)

Categories

FileIO

Description

Reads Verilog or SystemVerilog source files. This command is similar to the `add_files` command. The Verilog file is added to the source fileset as it is read. If the `-library` argument is specified, the file is added with the Library property defined appropriately.

You can use this command to read the contents of source files into the in-memory design, when running the Vivado tool in Non Project mode, in which there is no project file to maintain and manage the various project source files. Refer to the *Vivado Design Suite User Guide: Design Flows Overview* (UG892) for more information on Non Project mode.

Because SystemVerilog is a superset of the Verilog language, the `read_verilog` command can read both file types. However, for SystemVerilog files, the `-sv` option needs to be specified for `read_verilog` to enable compilation in the SystemVerilog mode. In this mode, the tool recognizes and honors the SystemVerilog keywords and constructs.

You can have a mixture of both Verilog files (.v files), and SystemVerilog files (.sv files), as well as VHDL (using `read_vhdl`). When the tool compiles these files for synthesis, it creates separate "compilation units" for each file type. All files of the same type are compiled together.

Arguments

`-library <arg>` - (Optional) The library the Verilog file should reference. The default Verilog library is xil_defaultlib. The library name is ignored by Vivado synthesis.

`-sv` -(Optional) Read the files as a SystemVerilog compilation group.

Note: Since Verilog is a subset of SystemVerilog, unless a Verilog source has user-defined names that collide with reserved SystemVerilog keywords, reading Verilog files with the -sv switch enables SystemVerilog compilation mode for those files. However, adding a SystemVerilog file in a Verilog compilation unit (without `-sv`) will not work.

`<files>` - (Required) The name of one or more Verilog files to be read.

Note: If the path is not specified as part of the file name, the tool will search for the specified file in the current working directory and then in the directory from which the tool was launched.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example reads the specified Verilog file and adds it to the source fileset:

```
read_verilog C:/Data/FPGA_Design/new_module.v
```

The following example creates two compilation units, one for SystemVerilog files and one for Verilog files:

```
read_verilog -sv { file1.sv file2.sv file3.sv }
read_verilog { file1.v file2.v file3.v}
```

See Also

- add_files
- read_vhdl
- remove_files

read_vhdl

Read one or more VHDL files

Syntax

```
read_vhdl -library <arg> [-vhdl2008] [-quiet] [-verbose] <files>
```

Returns

List of file objects that were added.

Usage

Name	Description
`-library`	VHDL library
`[-vhdl2008]`	VHDL file is version 2008.
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<files>`	VHDL file name(s)

Categories

FileIO

Description

Reads VHDL source files. This command is similar to the `add_files` command. The VHDL files are added to the source fileset as the file is read. If the `-library` argument is specified, the file is added with the Library property defined.

You can use this command to read the contents of source files into the in-memory design, when running the Vivado tool in Non Project mode, in which there is no project file to maintain and manage the various project source files. Refer to the *Vivado Design Suite User Guide: Design Flows Overview* (UG892) for more information on Non Project mode.

Arguments

`-library <arg>` - (Optional) The library the VHDL file should reference. The default VHDL library is `xil_defaultlib`.

`-vhdl2008` - (Optional) Identifies the files to be read as VHDL file version 2008.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Send Feedback

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<files>` - (Required) Names of one or more VHDL files to be read.

Note: If the path is not specified as part of the file name, the tool will search for the specified file in the current working directory and then in the directory from which the tool was launched.

Examples

The following example reads the specified VHDL file and adds it to the source fileset:

```
read_vhdl C:/Data/FPGA_Design/new_module.vhdl
```

This example reads multiple specified VHDL 2008 files:

```
read_vhdl -vhdl2008 {file1.vhd file2.vhd file3.vhd}
```

See Also

- add_files
- read_verilog
- remove_files

read_xdc

read physical and timing constraints from one of more files.

Syntax

```
read_xdc [-cells <args>] [-ref <arg>] [-quiet_diff_pairs] [-mode <arg>]
    [-unmanaged] [-no_add] [-quiet] [-verbose] <files>
```

Returns

List of files.

Usage

Name	Description
[-cells]	Import constraints for these cells
[-ref]	Import constraints for this ref
[-quiet_diff_pairs]	Suppress warnings about differential pair inference when importing I/O ports
[-mode]	Import constraints as out_of_context. Values: default, out_of_context Default: default
[-unmanaged]	treat this file as unmanaged constraints file
[-no_add]	don't add this file to constraints fileset
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<files>	Input file(s) to read

Categories

FileIO

Description

Imports physical and timing constraints from a Xilinx Design Constraints file (XDC). The XDC is imported into the `current_instance` level of the design hierarchy, which defaults to the top-level of the design, or can be imported into specified cells. When imported at the top-level, the specified XDC file is added to the active constraint fileset.

> ⭐ **IMPORTANT!** *Constraints from the XDC file will overwrite any current constraints of the same name. Therefore, exercise some caution when reading a XDC file to be sure you will not overwrite important constraints.*

This command is similar to the `add_files` command in that the XDC file is added by reference rather than imported into the local project directory.

You can use this command to read the contents of source files into the in-memory design, when running the Vivado tool in Non Project mode, in which there is no project file to maintain and manage the various project source files. Refer to the *Vivado Design Suite User Guide: Design Flows Overview* (UG892) for more information on Non Project mode.

Arguments

`-cells` <args> - (Optional) Apply the constraints from the XDC file to the specified instance names. The constraints will be applied ONLY to the specified cell instances, and the XDC file will not be added to the active constraint fileset.

> 💡 **TIP:** *A design must be open when specifying the -cells option.*

`-ref <arg>` - (Optional) Read the constraints from the XDC file and apply them to ALL instances of the referenced module, wherever they happen to be instantiated in the current design.

`-quiet_diff_pairs` - (Optional) Suppress warnings about differential pair inference when importing I/O constraints.

`-mode [ default | out_of_context ]` - (Optional) Import the specified constraint files as in-context with the top-level design, or as out_of_context constraints to be used when generating output products for hierarchical modules or IP cores. For more information on the out-of-context design flow, refer to the *Vivado Design Suite User Guide: Design Flows Overview (UG892).*

> ⭐ **IMPORTANT!** *Out-of-context constraints should be added to specified cells or cell instances.*

`-unmanaged` - (Optional) Treat the added files as unmanaged Tcl constraint files. The Vivado tool will not save constraint changes back into these unmanaged Tcl files. For more information on unmanaged Tcl constraints, refer to the *Vivado Design Suite User Guide: Using Constraints (UG903).*

`-no_add` - (Optional) Read the constraints from the file, and integrate them into the in-memory design, but do not add the XDC file to the list of files in the current constraint set.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<files>` - (Required) The filenames of the XDC files to be imported.

Note: If the path is not specified as part of the file name, the tool will search for the specified file in the current working directory and then in the directory from which the tool was launched.

Examples

The following example reads the XDC file and applies it to the current design:

```
read_xdc file_1.xdc
```

The following example reads the XDC file and applies it ALL instances of the referenced module found in the current design:

```
read_xdc -ref hex2led file_2.xdc
```

The following example reads the XDC file and applies it ONLY to the specified instance within the referenced module:

```
read_xdc -ref sixty -cells lsbcount file_3.xdc
```

The following example reads the XDC file and applies it to the specified instances in the current design, even though they are instances of different modules:

```
read_xdc -cells {one_decode sixty/msbcount} file_4.xdc
```

Note: Multiple cells must be enclosed in quotes, "", or braces, {}.

See Also

- add_files
- current_instance
- infer_diff_pairs
- write_xdc

readback_hw_cfgmem

Readback data from the hw_cfgmem object

Syntax

```
readback_hw_cfgmem [-checksum] [-force] [-all] [-offset <arg>] -file <arg>
    [-format <arg>] [-datacount <arg>] [-quiet] [-verbose] [<hw_cfgmem>...]
```

Usage

Name	Description
[-checksum]	readback and calculate checksum; cannot be used with -file option
[-force]	force write of file
[-all]	specify readback of all memory locations
[-offset]	memory offset value Default: 0x0
-file	File to write readback to
[-format]	File format of readback file
[-datacount]	number of data units to readback Default: 0x0
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<hw_cfgmem>]	list of hardware cfgmems Default: current hardware cfgmem

Categories

Hardware

Description

Read programming data off of the hardware configuration memory device, specified as a hw_cfgmem object.

This command reads back the memory configuration file data programmed into a flash memory device by the `program_hw_cfgmem` command and writes it to the specified file. The memory configuration file is created by the `write_cfgmem` command and combines the bitstream (`.bit`) file, and any specified data files, into the memory configuration file format.

Readback is the process of reading data from the configuration memory device to verify that the bitstream and any additional data files were properly programmed into the flash memory device.

Arguments

`-checksum` - (Optional) Calculate a checksum for the bitstream from the device.

Send Feedback

`-force` - (Optional) Force the overwriting of the specified file if one of the same name already exists.

`-all` - (Optional) Read back all the address locations on the configuration memory device.

> 💡 **TIP:** *By default only the addresses defined by the configuration memory file (PROGRAM.FILE) of the specified hw_cfgmem object will be read back, although this can be affected by the* `-offset` *and* `-datacount` *options.*

`-offset <arg>` - (Optional) Memory address offset value to begin reading back from. The default offset address is 0x0.

`-file <arg>` - (Required) Write the data read back from the hw_cfgmem object to the specified file. The readback file is similar to the MCS file created by the `write_cfgmem` command. The filename suffix should be either `.mcs` or `.bin` to reflect the format of the contents.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-format [ mcs | bin ]` - (Optional) File format of the readback file to create. The default format is MCS.

`-datacount <arg>` - (Optional) Specify the number of bytes to read back. The default is to read all data starting from the address specified by the `-offset` option.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_cfgmem> - (Optional) The hw_cfgmem object to read the data back from. The hw_cfgmem must be specified as an object as returned by the `get_hw_cfgmems` or `current_hw_cfgmem` commands. If the hw_cfgmem is not specified, the current_hw_cfgmem will be used.

Example

The following example creates a hw_cfgmem object associated with the `current_hw_device`; sets a property defining the memory configuration file (PROGRAM.FILE) previously created from the bitstream with the `write_cfgmem` command; sets other properties of the hw_cfgmem object for use during the readback process; and programs the current hw_device with the cfgmem bitstream:

```
create_hw_cfgmem -hw_device [current_hw_device] \
    [lindex [get_cfgmem_parts {n25q128-3.3v-spi-x1_x2_x4}] 0]
set cfgMem [current_hw_cfgmem]
set_property PROGRAM.FILE {C:/Data/config_n25q128.mcs} $cfgMem
set_property PROGRAM.ADDRESS_RANGE  {use_file} $cfgMem
set_property PROGRAM.BLANK_CHECK  1 $cfgMem
set_property PROGRAM.ERASE  1 $cfgMem
set_property PROGRAM.CFG_PROGRAM  1 $cfgMem
set_property PROGRAM.VERIFY  1 $cfgMem
create_hw_bitstream -hw_device [current_hw_device] \
    [get_property PROGRAM.HW_CFGMEM_BITFILE [current_hw_device]]
program_hw_devices [current_hw_device]
```

Note: The hw_cfgmem object is assigned to the Tcl variable $cfgMem.

The following example reads back the current hw_cfgmem object using the addresses defined in the object's PROGRAM.FILE property:

```
readback_hw_cfgmem -format mcs \
    -file C:/Data/design1.mcs [current_hw_cfgmem]
```

The following example reads back all the addresses from the current hw_cfgmem object, starting at address 0 and up to the maximum memory depth:

```
readback_hw_cfgmem -all -format mcs \
    -file C:/Data/design1.mcs [current_hw_cfgmem]
```

The following example reads back a select range of addresses from the current hw_cfgmem object:

```
readback_hw_cfgmem -offset 0x084 -datacount 100 -format mcs \
    -file C:/Data/design1.mcs [current_hw_cfgmem]
```

See Also

- create_hw_cfgmem
- current_hw_cfgmem
- delete_hw_cfgmem
- get_cfgmem_parts
- get_hw_cfgmems
- get_property
- program_hw_cfgmem

- write_cfgmem

Send Feedback

readback_hw_device

Readback hardware devices

Syntax

```
readback_hw_device [-force] [-capture] [-readback_file <arg>]
    [-bin_file <arg>] [-quiet] [-verbose] [<hw_device>...]
```

Returns

Hardware devices

Usage

Name	Description
[-force]	force write of file
[-capture]	capture configuration readback data (ultrascale only)
[-readback_file]	readback file for rbd file output
[-bin_file]	bin file for bin file output
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<hw_device>]	list of hardware devices Default: current hardware device

Categories

Hardware

Description

Read bitstream data from the current hardware device and write it to the specified readback or binary file.

The Vivado device programmer will readback bitstream data from the Xilinx device through the open target.

> **IMPORTANT!** *If the bitstream on the hw_device is encrypted, readback is not permitted.*

This command returns the name of the readback file created, or returns an error if it fails.

Arguments

`-force` - (Optional) Force the overwriting of the specified file if one of the same name already exists.

`-capture` - (Optional) Enable readback capture of configuration data on UltraScale devices only. Readback capture can be used to determine the content of user state elements, such as CLB registers, block RAM, distributed RAM, and SRL contents.

`-readback_file <arg>` - (Optional) Write a readback file (RDB) which is an ASCII output of the bitstream read from the specified hw_device. This file is similar to the ASCII file that is produced by the `write_bitstream -raw_bitfile` command, except that it has no header.

> **IMPORTANT!** *Although both `-readback_file` and `-bin_file` are optional, one or both of these options must be specified or the command has no output file to write the results to.*

`-bin_file <arg>` - (Optional) Write a binary file (BIN) containing the programming data read back from the specified hw_device. This file is similar to the binary file that is produced by the `write_bitstream -bin_file` command.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<hw_device>` - (Optional) The hw_device object to read the programming data from. The hw_device must be specified as an object as returned by the `get_hw_devices` or `current_hw_device` commands. If the hardware device is not specified, the `current_hw_device` will be used.

Example

The following example writes an ASCII file of the bitstream data read back from the current hardware device:

```
readback_hw_device -readback_file C:/Data/readback_1.rbd [current_hw_device]
```

See Also

- connect_hw_server
- create_hw_cfgmem
- current_hw_device
- get_property

- open_hw_target
- program_hw_devices
- verify_hw_devices
- write_bitstream

redo

Re-do previous command

Syntax

```
redo [-list] [-quiet] [-verbose]
```

Returns

With -list, the list of redoable tasks.

Usage

Name	Description
[-list]	Show a list of redoable tasks
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

GUIControl

Description

> ⭐ **IMPORTANT!** *The `UNDO` and `REDO` commands are intended for use in the Vivado IDE, and are not recommended for use in Tcl scripts to restore designs to a former state. To restore a design to a specific condition, you must write a design checkpoint using the `write_checkpoint` command, to be restored using `read_checkpoint`.*

Redo a command that has been previously undone. This command can be used repeatedly to redo a series of commands.

If a command group has been created using the `startgroup` and `endgroup` commands, the redo command will redo the group of commands as a sequence.

Arguments

`-list` - (Optional) Get the list of commands that can be redone. When you use the `undo` command, the tool will step backward through a list of commands. The `redo` command can then be used to redo those commands.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Send Feedback

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example returns a list of commands that can be redone:

```
redo -list
```

See Also

- undo
- startgroup
- endgroup

Send Feedback

refresh_design

Refresh the current design

Syntax

```
refresh_design [-part <arg>] [-quiet] [-verbose]
```

Usage

Name	Description
[-part]	Target part
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Project](#)

Description

Reloads the current design from the project data on the hard drive. This overwrites the in-memory view of the design to undo any recent design changes.

Arguments

`-part <arg>` - (Optional) The new target part for the design when it is reloaded. This overrides the constraint file part specified in the project data on the hard drive.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Send Feedback

Examples

The following command reloads the current design from the project data on hard disk. This will overwrite the unsaved changes of the design which are in memory.

```
refresh_design
```

Note: You can use the command to undo a series of changes to the design and revert to the previously saved design.

The following example refreshes the current design using the specified V6 part as the target device. The second command is required to make the selected part the target device for the active implementation run.

```
refresh_design -part xc6vcx75tff784-1
set_property part xc6vcx75tff784-1 [get_runs impl_6]
```

Note: The second command is not required if the target part is not changed.

See Also

- set_property

refresh_hw_axi

Refresh hardware AXI object status.

Syntax

```
refresh_hw_axi [-quiet] [-verbose] [<hw_axis>...]
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<hw_axis>]	List of hardware AXI objects.

Categories

Hardware

Description

Refresh the STATUS properties of the hw_axi object with the values from the current hw_device.

The refresh command takes the values from the status registers of the JTAG to AXI MASTER on the hardware device, and populates them into the appropriate properties of the hw_axi object in the hardware manager.

Refresh the STATUS properties of the specified hw_axi objects. THE STATUS properties include: STATUS.AXI_READ_BUSY, STATUS.AXI_READ_DONE, STATUS.AXI_WRITE_BUSY, STATUS.AXI_WRITE_DONE, STATUS.BRESP, and STATUS.RRESP.

This command updates the properties on the hw_axi object, but otherwise returns nothing if successful. The command returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Send Feedback

Note: Message limits can be defined with the `set_msg_config` command.

<hw_axis> - (Required) The hw_axi objects to refresh. The hw_axi must be specified as an object returned by the `get_hw_axis` command.

Example

The following example refreshes the STATUS properties of the hw_axi objects:

```
refresh_hw_axi [get_hw_axis]
```

See Also

- create_hw_axi_txn
- delete_hw_axi_txn
- get_hw_axis
- get_hw_axi_txns
- reset_hw_axi

refresh_hw_ddrmc

Refresh the status of the current hardware object. Inputs need to be any DDRMC object. At least one object is required. If properties are specified, but do not exist in the object, those properties will not be refreshed.

Syntax

```
refresh_hw_ddrmc [-regexp] [-properties <args>] [-quiet] [-verbose]
    <hw_objects>
```

Usage

Name	Description
[-regexp]	Properties list contains full regular expressions
[-properties]	List of properties to refresh Default: All properties in object
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hw_objects>	hardware DDRMC objects

Categories

[Hardware](#)

Description

Refreshes the in-memory contents of all of the properties, or specified properties, of the specified Versal DDR Memory Controller (DDRMC) objects with values read from the current hardware device.

The refresh command takes the values from the DDRMC object implemented on the hardware device, and populates them into the appropriate properties of the hw_ddrmc object in the Vivado hardware manager. If only certain properties are specified, then those properties are updated from the hw device.

At least one object is required. If properties are specified that do not exist in the object, that property will not be refreshed.

This command updates the properties on the hw_ddrmc object, but otherwise returns nothing if successful. The command returns an error if it fails.

Arguments

-regexp - (Optional) The list of properties to refresh is defined using regular expression.

Send Feedback

`-properties` <args> - (Optional) Refresh the specified property or properties of the hw_ddrmc object. As a default behavior, if no properties are specified, all properties of the specified object or objects will be refreshed from the current values on the hardware device.

> **TIP:** *If a non-existent property is specified, that property is ignored.*

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_objects> - (Required) Specifies the specific DDRMC objects (hw_ddrmc) to run.

Examples

The following example refreshes all of the properties of the DDRMC objects in the Vivado hardware manager with the properties from the current hw_device:

```
refresh_hw_ddrmc -properties {PHY_RANKS} [lindex [get_hw_ddrmcs] 3]
```

See Also

- current_hw_device
- current_hw_target
- get_hw_ddrmcs
- report_hw_ddrmc

refresh_hw_device

Refresh a hardware device. Read device and core information from device.

Syntax

```
refresh_hw_device [-update_hw_probes <arg>] [-disable_done_check]
    [-force_poll] [-quiet] [-verbose] [<hw_device>]
```

Usage

Name	Description
[-update_hw_probes]	Update hardware probe information, read from probes file
[-disable_done_check]	Disable done check for refresh device
[-force_poll]	Force poll of all targets
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<hw_device>]	hardware device Default: current hardware device

Categories

Hardware

Description

Refreshes the in-memory view of the device by scanning for debug and IBERT cores on the specified hw_device object, and also reads a probe file when directed.

The Hardware Manager in the Vivado Design Suite creates, deletes, or updates the hw_ila, hw_vio, hw_sio*, and hw_axi objects based on the core information found in the device, and also what is read from the probes file in the case of ILA and VIO debug cores.

Use the `refresh_hw_device` after the `program_hw_devices` to keep the in-memory hardware debug objects in sync with the state of the actual cores on the physical device.

Arguments

`-update_hw_probes <arg>` - (Optional) Update the probes file (`.ltx`) associated with the specified device by reading the specified probes file.

Note: The probes file is associated with a hw_device object through the use of the `set_property` command to define the PROBES.FILE property.

Send Feedback

`-disable_done_check` - (Optional) Disable the DONE check to enable the hardware manager to scan for debug cores on the current device, even if the DONE pin status is low. A High signal on the DONE pin indicates completion of the configuration sequence. The DONE check waits for the high state on the DONE pin to confirm configuration is complete.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_device> - (Optional) Specify the hw_device object to refresh. The hw_device must be specified as an object as returned by the `get_hw_devices` or the `current_hw_device` commands. If the hardware device is not specified, the `current_hw_device` will be returned.

Example

The following example refreshes the specified hw_device:

```
refresh_hw_device [lindex [get_hw_devices] 0]
```

See Also

- current_hw_device
- get_hw_devices
- program_hw_devices

refresh_hw_hbm

Refresh the status of the current hardware object. Inputs can be HBM or device hardware object. At least one object is required. If properties are specified that do not exist in the object, that property will not be refreshed.

Syntax

```
refresh_hw_hbm [-regexp] [-properties <args>] [-quiet] [-verbose]
    <hw_objects>
```

Usage

Name	Description
`[-regexp]`	Properties list contains full regular expressions
`[-properties]`	List of properties to refresh Default: All properties in object
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<hw_objects>`	hardware objects

Categories

Hardware

Description

Refreshes the in-memory view of all of the properties, or specified properties, of the specified HBM controller objects with values read from the current hardware device.

The refresh command takes the values from the HBM controller implemented on the hardware device, and populates them into the appropriate properties of the hw_hbm object in the Vivado hardware manager. If only certain properties are specified, then those properties are updated from the hw device.

At least one object is required. If properties are specified that do not exist in the object, that property will not be refreshed.

This command updates the properties on the hw_hbm object, but otherwise returns nothing if successful. The command returns an error if it fails.

Arguments

`-regexp` - (Optional) The list of properties to refresh is defined using regular expression.

Send Feedback

`-properties` <args> - (Optional) Refresh the specified property or properties of the hw_hbm object. As a default behavior, if no properties are specified, all properties of the specified object or objects will be refreshed from the current values on the hardware device.

💡 **TIP:** *If a non-existent property is specified, that property is ignored.*

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_objects> - (Required) Specifies the specific HBM activity monitor objects (hw_hbms) to run.

Examples

The following example refreshes all of the properties of the HBM controller in the Vivado hardware manager with the properties from the current hw_device:

```
refresh_hw_hbm [lindex [get_hw_hbms] 1]
```

The following example refreshes the specified property of the HBM controller from the current hw_device:

```
refresh_hw_hbm -properties {MC2.INIT.AM_REPEAT_EN} [get_hw_hbms *HBM_2]
```

See Also

- add_hw_hbm_pc
- commit_hw_hbm
- current_hw_device
- current_hw_target
- get_hw_hbms
- pause_hw_hbm_amon
- remove_hw_hbm_pc
- resume_hw_hbm_amon
- run_hw_hbm_amon
- stop_hw_hbm_amon

refresh_hw_mig

Refresh the status of the current hardware object. Inputs can be any mig, device, target, or server hardware object. At least one object is required. If properties are specified that do not exist in the object, that property will not be refreshed.

Syntax

```
refresh_hw_mig [-regexp] [-properties <args>] [-quiet] [-verbose]
    <hw_objects>
```

Usage

Name	Description
[-regexp]	Properties list contains full regular expressions
[-properties]	List of properties to refresh Default: All properties in object
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hw_objects>	hardware objects

Categories

Hardware

Description

Refreshes the in-memory view of all of the properties, or specified properties, of the specified hw_mig objects with values read from the current hardware device.

The refresh command takes the values from the memory controller implemented on the hardware device, and populates them into the appropriate properties of the hw_mig debug core in the Vivado logic analyzer, or standalone Vivado Lab Edition.

At least one object is required. If properties are specified that do not exist in the object, that property will not be refreshed.

This command updates the properties on the hw_mig object, but otherwise returns nothing if successful. The command returns an error if it fails.

Arguments

`-regexp` - (Optional) The list of properties to refresh is defined using regular expression.

Send Feedback

`-properties` <args> - (Optional) Refresh the specified property or properties of the hw_mig object. As a default behavior, if no properties are specified, all properties of the specified object or objects will be refreshed from the current values on the hardware device.

💡 **TIP:** *If a non-existent property is specified, that property is ignored.*

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_objects> - (Optional) Inputs can be any hw_mig, hw_device, hw_target, or hw_server object.

Note: The objects must be specified using the appropriate `get_hw_xxx` command, for instance `get_hw_migs`, rather than specified by name.

Example

The following example refreshes all of the properties of the memory IP in the Vivado logic analyzer with the properties from the current hw_device:

```
refresh_hw_mig [lindex [get_hw_migs] 0]
```

See Also

- commit_hw_mig
- connect_hw_server
- current_hw_device
- get_hw_migs
- implement_mig_cores
- report_hw_mig
- set_property

Send Feedback

refresh_hw_pcie

Refresh the PCIe core properties, by reading from HW.

Syntax

```
refresh_hw_pcie [-quiet] [-verbose] <hw_pcie>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hw_pcie>	Hardware PCIe object

Categories

Hardware

Description

Refresh for the Peripheral Component Interconnect Express (PCIe) debug core object, hw_pcie, defined on the current hardware device.

The customizable LogiCORE IP PCIe core for Xilinx ACAPs is designed for evaluating and monitoring the PCIe Link Training and Status State Machine (LTSSM) running on the Gigabit Transceivers (GTs). In the Vivado Hardware Manager, Versal PCIe soft cores implemented in the design, are represented as hw_pcie objects. You can use these PCIe debug cores to solve a range of debug and validation problems; from viewing the PCIe link information to the LTSSM state transition diagram.

This command reads data from the PCIe debug core and updates the relevant properties in the hw_pcie object.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<hw_pcie>` - (Required) List of HW PCIe objects

Examples

The following example refreshes the PCIe core at index 0

```
refresh_hw_pcie [lindex [get_hw_pcies] 0]
```

See Also

- get_hw_pcies
- report_hw_pcie
- reset_hw_pcie

Send Feedback

refresh_hw_server

Refresh a connection to a hardware server

Syntax

```
refresh_hw_server [-force_poll] [-quiet] [-verbose] [<hw_server>]
```

Usage

Name	Description
[-force_poll]	Force poll of all targets
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<hw_server>]	hardware server

Categories

Hardware

Description

Refresh or reopen the connection to the current or specified hardware server.

This command returns the connection messages from the hardware server, or returns an error if it fails.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the set_msg_config command.

<hw_server> - (Optional) The hw_server to refresh. If the server is not specified, the current_hw_server will be refreshed. The hardware server can be specified by name, or specified as a hw_server object returned by the get_hw_servers or current_hw_server commands.

Send Feedback

Example

The following example refreshes the current_hw_server:

```
refresh_hw_server
```

The following example refreshes all connected hardware servers, printing the host name prior to refreshing th server:

```
foreach x [get_hw_server] {puts "Refreshing Host $x"; refresh_hw_server $x}
```

See Also

- connect_hw_server
- current_hw_server
- disconnect_hw_server
- get_hw_servers

Send Feedback

refresh_hw_sio

Refresh the status of the specified hardware objects. Inputs can be any serial I/O (except scan and sweep), device, target, or server hardware object. At least one object is required. If properties are specified that do not exist in the object, that property will not be refreshed.

Syntax

```
refresh_hw_sio [-regexp] [-properties <args>] [-quiet] [-verbose]
    <hw_objects>
```

Usage

Name	Description
`[-regexp]`	Properties list contains full regular expressions
`[-properties]`	List of properties to refresh Default: All properties in object
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<hw_objects>`	hardware objects

Categories

[Hardware](#)

Description

Refreshes the in-memory view of all of the properties, or specified properties, of the specified hw_sio objects with values read from the actual object on the hardware device.

Specified objects can include any serial I/O object such as GTs, RXs, TXs, PLLs, or Commons, excluding hw_sio_scan and hw_sio_sweep objects. SIO objects also include device, target, or server hardware objects.

The `refresh_hw_sio` command reads the values of the specified objects on the hardware device, and applies the value to the associated property of the IBERT core in the Hardware Manager.

This command returns no feedback of its operation if successful, or returns an error if it fails.

Arguments

`-regexp` - (Optional) The list of properties to refresh is defined using regular expression.

`-properties` <args> - (Optional) Refresh the specified property or properties of the hw_sio objects. As a default behavior, if no properties are specified, all properties of the specified object or objects will be refreshed from the current values on the hardware device.

💡 **TIP:** *If a non-existent property is specified, that property is ignored.*

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_objects> - (Required) Specify one or more hw_sio objects to refresh. The hw_sio objects must be specified as objects returned by one of the `get_hw_sio_*` commands.

Example

The following example refreshes all properties on the specified hw_sio_gt object:

```
refresh_hw_sio [get_hw_sio_gts *MGT_X0Y11]
```

The following example refreshes all of the properties on all of the hw_sio objects on the current hardware device:

```
refresh_hw_sio [current_hw_device]
```

See Also

- current_hw_device
- get_hw_devices
- get_hw_servers
- get_hw_sio_commons
- get_hw_sio_gts
- get_hw_sio_iberts
- get_hw_sio_links
- get_hw_sio_plls
- get_hw_sio_txs
- get_hw_targets
- report_property

refresh_hw_softmc

Refresh the status of the current hardware object. Inputs need to be any Soft MC object. At least one object is required. If properties are specified, but do not exist in the object, those properties will not be refreshed.

Syntax

```
refresh_hw_softmc [-regexp] [-properties <args>] [-quiet] [-verbose]
    <hw_objects>
```

Usage

Name	Description
[-regexp]	Properties list contains full regular expressions
[-properties]	List of properties to refresh Default: All properties in object
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hw_objects>	hardware Soft MC objects

Categories

Hardware

Description

Refreshes the in-memory contents of all of the properties, or specified properties, for the specified Versal Soft Memory Controller (SoftMC) objects with values read from the current hardware device. The refresh command takes the values from the SoftMC object implemented on the hardware device, and populates them into the appropriate properties of the `hw_softmc` object in the Vivado hardware manager. If only certain properties are specified, then those properties are updated from the hw device.

At least one object is required. If properties are specified as arguments, but do not exist in the object, those properties will not be refreshed. This command updates the properties on the `hw_softmc` object, but otherwise returns nothing if successful. The command returns an error if it fails.

Arguments

`-regexp` - (Optional) The list of properties to refresh is defined using regular expression.

`-properties <arg>` - (Optional) Refresh the specified property or properties of the `hw_softmc` object. As a default behavior, if no properties are specified, all properties of the specified object or objects will be refreshed from the current values on the hardware device.

Note: If a non-existent property is specified, that property is ignored.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<hw_objects>` - (Required) Specifies the specific Soft MC objects (`hw_softmc`) to run.

Examples

The following example refreshes all of the properties on the fourth SoftMC object found in the Vivado hardware manager with the properties from the current `hw_device`:

```
refresh_hw_softmc [lindex [get_hw_softmcs] 3]
```

See also

- current_hw_device
- current_hw_target
- get_hw_softmc
- report_hw_softmc

Send Feedback

refresh_hw_sysmon

Refresh the status of the current hardware object. Inputs can be hw_server, hw_target, hw_device or hw_sysmon objects. At least one object is required. If properties are specified that do not exist in the object, that property will not be refreshed.

Syntax

```
refresh_hw_sysmon [-regexp] [-properties <args>] [-quiet] [-verbose]
    <hw_objects>
```

Usage

Name	Description
`[-regexp]`	Properties list contains full regular expressions
`[-properties]`	List of properties to refresh Default: All properties in object
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<hw_objects>`	hardware objects

Categories

Hardware

Description

Refresh the properties of the hw_sysmon object with the values on the system monitor (XADC) from the current hw_device.

The refresh command takes the values from the status registers of the system monitor on the hardware device, and populates them into the appropriate properties of the hw_sysmon object in the hardware manager.

> **TIP:** *The hw_sysmon object is automatically refreshed at the rate specified by the SYSMON_REFRESH_RATE_MS on the object.*

This command updates the properties on the hw_sysmon object, but otherwise returns nothing if successful. The command returns an error if it fails.

Arguments

`-regexp` - (Optional) The list of properties to refresh is defined using regular expression.

`-properties` <args> - (Optional) Refresh the specified property or properties of the hw_sysmon object. As a default behavior, if no properties are specified, all properties of the specified object or objects will be refreshed from the current values on the hardware device.

> **TIP:** *If a non-existent property is specified, that property is ignored.*

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_objects> - (Optional) Specify the system monitor to refresh the properties of, with values from the hw_device. The system monitor object can be specified as the hw_sysmon object, or as the system monitor through the associated hw_device, hw_target, or hw_server objects.

Note: The objects must be specified using the appropriate `get_hw_xxx` command, for instance `get_hw_sysmon`, rather than specified by name.

Example

The following example refreshes the TEMPERATURE property of the hardware system monitor object with the actual temperature on the current device:

```
refresh_hw_sysmon -properties {TEMPERATURE} [lindex [get_hw_sysmons] 0]
```

See Also

- commit_hw_sysmon
- connect_hw_server
- current_hw_device
- get_hw_sysmons
- set_hw_sysmon_reg
- set_property

refresh_hw_target

Refresh a hardware target

Syntax

```
refresh_hw_target [-force_poll] [-quiet] [-verbose] [<hw_target>]
```

Usage

Name	Description
[-force_poll]	Force poll of all targets
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<hw_target>]	hardware target

Categories

Hardware

Description

Refresh the connection to the specified hardware target on the current hardware server, and reload the hw_target object in the Hardware Manager of the Vivado Design Suite. If no hw_target object is specified, the `current_hw_target` will be refreshed.

The hardware target is a system board containing a JTAG chain of one or more Xilinx devices that you can program with a bitstream file, or use to debug your design. Connections between hardware targets on the system board and the Vivado Design Suite are managed by the Xilinx hardware server application, and the `connect_hw_server` command. Refer to *Vivado Design Suite User Guide: Programming and Debugging (UG908)* for a list of supported JTAG download cables and devices.

Each hardware target can have one or more Xilinx devices to program, or to use for debugging purposes. The current device is specified or returned by the `current_hw_device` command. After specifying the current hardware target, you can open the connection through the hardware target, to the Xilinx FPGA using the `open_hw_target` command.

`refresh_hw_target` scans the devices on the hardware target and creates, deletes, or updates the hw_device objects available through the target. Available devices are returned using the `get_hw_devices` command.

This command returns a transcript of the refresh process, or returns an error if it fails.

Send Feedback

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_target> - (Optional) Specify the hw_target object to refresh the connection to. The hw_target must be specified as an object as returned by the `get_hw_targets` or `current_hw_target` commands. If no target is specified, the Vivado tool will refresh the connection to the `current_hw_target`.

Example

The following example refreshes the current hardware target:

```
refresh_hw_target
```

See Also

- connect_hw_server
- current_hw_device
- current_hw_target
- get_hw_devices
- get_hw_targets
- get_hw_servers
- open_hw_target

Send Feedback

refresh_hw_vio

Update hardware probe INPUT_VALUE and ACTIVITY_VALUE properties with values read from hardware VIO core(s).

Syntax

```
refresh_hw_vio [-update_output_values] [-quiet] [-verbose] <hw_vios>...
```

Usage

Name	Description
`[-update_output_values]`	Update hardware probe OUTPUT_VALUE property with values read from VIO core(s).
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<hw_vios>`	List of hardware VIO objects.

Categories

Hardware

Description

Update the INPUT_VALUE and ACTIVITY_VALUE properties of the input probes of the specified VIO debug cores with values read from the hw_vio core on the hardware device.

The Virtual Input/Output (VIO) debug core can both monitor and drive internal signals on a programmed Xilinx device in real time. The VIO core uses hardware probes, hw_probe objects, to monitor and drive signals on the device. Input probes monitor signals as inputs to the VIO core. Output probes drive signals to specified values from the VIO core.

The `refresh_hw_vio` command reads the signal values at the input probes of the VIO debug core on the device, and applies the value to the INPUT_VALUE property of the hw_probe, and updates the ACTIVITY_VALUE property on the probe as well.

This command returns nothing if successful, or returns an error if it fails.

Arguments

`-update_output_values` - (Optional) Update hardware probe OUTPUT_VALUE property with values read from the signals at the specified VIO debug cores. This is a boolean argument enabled by its presence. This option has the affect of resetting the OUTPUT_VALUE on the probe to match the signal value on the hw_vio debug core.

> 💡 **TIP:** *This is the reverse of the* `commit_hw_vio` *command, which applies the OUTPUT_VALUE on the probe to the hw_vio debug core on the device.*

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_vios> - (Required) Specify one or more hw_vio objects to refresh. The hw_vio objects can either be specified as objects returned by the `get_hw_vios` command, or specified by name.

Example

The following example refreshes the specified hw_vio debug core, specified by name, including updating the OUTPUT_VALUE property on the hw_probes:

```
refresh_hw_vio -update_output_values hw_vio_1
```

See Also

- commit_hw_vio
- connect_hw_server
- current_hw_device
- get_hw_probes
- get_hw_vios
- program_hw_devices
- reset_hw_vio_activity
- reset_hw_vio_outputs
- set_property

Send Feedback

refresh_meminit

Update and initialize the BRAM initialization strings with contents of elf files.

Syntax

```
refresh_meminit [-quiet] [-verbose]
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

FileIO, Project

regenerate_bd_layout

Regenerate layout.

Syntax

```
regenerate_bd_layout [-hierarchy <arg>] [-layout_file <arg>] [-routing]
    [-quiet] [-verbose]
```

Usage

Name	Description
[-hierarchy]	Hierarchy path to the window
[-layout_file]	layout file previously exported by write_bd_layout using native format
[-routing]	Preserve placement of blocks and regenerate routing
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

IPIntegrator

Description

Regenerate the layout of the current IP integrator subsystem design in the open canvas. This command updates and redraws the graphical elements of the subsystem design in the Vivado IDE.

Arguments

-hierarchy <arg> - (Optional) Specify a hierarchical module to regenerate. Use the get_bd_cells command to specify the hierarchical module as an object.

-layout_file <arg> - (Optional) Specify a native format block design layout file that was written by the write_bd_layout command.

-routing - (Optional) Refresh the routing in the IP integrator canvas, but do not refresh the placement of objects.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Example

The following example refreshes the IP integrator canvas in the Vivado IDE:

```
regenerate_bd_layout
```

The following example refreshes the specified hierarchical module in the current block design:

```
regenerate_bd_layout -hierarchy [get_bd_cell myHier1]
```

See Also

- current_bd_design
- open_bd_design
- start_gui
- write_bd_layout

register_proc

Register a Tcl proc with Vivado.

Syntax

```
register_proc [-quiet] [-verbose] <proc> [<tasknm>]
```

Returns

Nothing

Usage

Name	Description
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<proc>`	Name of proc to register. Proc must be known to Tcl
`[<tasknm>]`	Name of Tcl task that wraps the proc. Default: Register the proc using the root name proc (no namespaces).

Categories

Tools

Description

Register a Tcl procedure (proc) with the Vivado Tcl command interpretor to register the command with the Vivado Design Suite help system.

The following is an example Tcl proc defined for use with the Vivado Design Suite:

```
proc findCommand {option} {
  # Summary:
  # Searches through all Vivado Tcl commands for commands implementing
  #   the specified argument.
  # Argument Usage:
  # option: Specifies the argument to search for.
  # Return Value:
  # Returns a list of Tcl commands that implement the option.
  # Categories: personal
  foreach cmd [lsort [info commands *]]
  {
      catch {
          if {[regexp "$option" [help -syntax $cmd]]}
  {
              puts $cmd
```

Send Feedback

```
            }
        }
    }
} ;
# End
```

The commented lines beginning with '#' are used to define the help text for the registered command in the Vivado Design Suite help system.

- # Summary: provides a brief description of the command.

- # Argument Usage: provides a list and description of the various arguments for the proc.

- # Return Value: provides a description of what is returned by the proc.

- # Categories: provides an ability to define categories for registered procedures.

After registering the procedure as a Tcl command, the Vivado help system will return this text when queried with:

```
tasknm -help
-or-
help tasknm
```

This command returns the name of the registered proc.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<proc> - (Required) The name of the Tcl procedure loaded into the current Vivado Design Suite session. The Tcl proc must be defined and loaded into the Vivado Design Suite prior to registration.

<tasknm> - (Optional) Specify the Tcl command name that wraps the proc, for use in the Vivado Design Suite. Default: Register the proc using the root name proc.

Example

The following example registers a Tcl proc called findCommand as a Tcl command named findCmd:

```
register_proc findCommand findCmd
```

See Also

- unregister_proc

reimport_files

Reimport files when they are found out-of-date

Syntax

```
reimport_files [-force] [-quiet] [-verbose] [<files>...]
```

Returns

List of file objects that were imported.

Usage

Name	Description
[-force]	Force a reimport to happen even when the local files may be newer
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<files>]	List of files to reimport. If no files are specified, all files in the project that are out-of-date, will be reimported

Categories

Project

Description

Reimports project files. This updates the local project files from the original referenced source files.

Arguments

`-force` - (Optional) Reimport files even when the local project files may be newer than their referenced source files.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Send Feedback

`<files>` - (Optional) List of files to reimport. If no files are specified, all files in the project that are out-of-date, will be reimported. If you use `-force` and specify no files, all files in the project will be reimported.

Examples

The following example reimports all project files regardless of whether they are out of date, or the local files are newer than the referenced source file:

```
reimport_files -force
```

Note: No warnings will be issued for newer local files that will be overwritten.

The following example reimports the specified files to the project, but only if the original source file is newer than the local project file:

```
reimport_files C:/Data/FPGA_Design/source1.v \
    C:/Data/FPGA_Design/source2.vhdl
```

See Also

- add_files
- import_files

relaunch_sim

Recompile the design without changing compilation options and restart the current simulation

Syntax

```
relaunch_sim [-quiet] [-verbose]
```

Returns

Current simulation object.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Simulation

Description

Relaunch the simulator to perform analysis and verification of an updated design.

The `relaunch_sim` command suspends the current simulation, recompiles the current design into a new simulation snapshot, then connects the current simulation to the new snapshot, and restarts the simulation.

In the typical HDL debug cycle you will compile a design into a simulation snapshot and launch a simulation, configuring the Vivado simulator IDE to display the signals of interest in the waveform viewer, as well as the scopes and objects of interest. During the debug process you may discover issues with your code or test bench, make corrections to your design, recompile and relaunch the simulator.

This command lets you recompile the design, and relaunch the simulator while preserving the current Vivado simulator configuration, such as open waveform and code windows, Scopes and Objects window settings.

> ⭐ **IMPORTANT!** *The `relaunch_sim` command applies only to simulations running in the Vivado Design Suite IDE, not stand-alone or batch Vivado simulator runs.*

This command returns a transcript of its process, or returns an error if it fails.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the set_msg_config command.

Examples

The following command relaunches the current simulation:

```
relaunch_sim
```

See Also

- close_sim
- current_sim
- launch_simulation
- xsim

remove_bps

Remove breakpoints from a simulation

Syntax

```
remove_bps [-all] [-file <arg>] [-line <arg>] [-quiet] [-verbose]
    [<BreakPointObjsOrIds>...]
```

Usage

Name	Description
[-all]	Remove all breakpoints
[-file]	The specific file to remove the breakpoint from given a line number
[-line]	The specific line number to remove the breakpoint given a filename Default: -1
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<BreakPointObjsOrIds>]	A list of one or more breakpoint objects and/or breakpoint object ID's to be removed

Categories

[Simulation](#)

Description

Remove specified breakpoints from the current simulation. You must have an open simulation to use this command.

A breakpoint is a user-determined stopping point in the source code used for debugging the design. When simulating a design with breakpoints, simulation of the design stops at each breakpoint to let you examine values and verify the design behavior.

The breakpoints in the current simulation can be reported using the `report_bps` command.

This command returns nothing, or an error if the command fails.

Arguments

`-all` - (Optional) Remove all breakpoints in the current simulation.

> ⭐ **IMPORTANT!** *This option will remove ALL breakpoints without warning, even if other options are specified.*

`-file <arg>` - (Optional) Remove an existing breakpoint in the specified HDL source file.

`-line <arg>` - (Optional) Remove an existing breakpoint at the specified line number of the HDL source file.

Note: Both `-file` and `-line` must be used together to define an existing breakpoint.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<BreakPointObjsOrIds> - (Optional) Specifies one or more breakpoint objects in the current simulation to remove. The breakpoint object is returned by the `add_bp` command when the breakpoint is added to the simulation.

Examples

The following example removes all the breakpoints in the current simulation:

```
remove_bps -all
```

See Also

- add_bp
- report_bps

remove_cell

Remove cells from the current design

Syntax

```
remove_cell [-quiet] [-verbose] <cells>...
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<cells>	List of cells to remove

Categories

Netlist

Description

Remove cells from the current netlist in either an open Synthesized or Implemented design.

Note: You cannot remove cells from library macros, also called macro-primitives.

Netlist editing changes the in-memory view of the netlist in the current design. It does not change the files in the source fileset, or change the persistent design on the disk. Changes made to the netlist may be saved to a design checkpoint using the `write_checkpoint` command, or may be exported to a netlist file such as Verilog, VHDL, or EDIF, using the appropriate `write_*` command.

Note: Netlist editing is not allowed on the elaborated RTL design.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<cells>` - (Required) List of cells to remove. The instance name can be specified as a hierarchical name, from the top-level of the design. In this case, you must use the hierarchy separator character in the hierarchical instance name. You can determine the current hierarchy separator with the `get_hierarchy_separator` command.

Examples

The following example removes the fftEngine from the in-memory netlist of the current design:

```
remove_cell fftEngine
remove_cell usbEngine0/usb_out
```

See Also

- create_cell
- write_checkpoint
- write_edif
- write_verilog
- write_vhdl

Send Feedback

remove_cells_from_pblock

Remove cells from a Pblock

Syntax

```
remove_cells_from_pblock [-quiet] [-verbose] <pblock> <cells>...
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<pblock>	Pblock to remove cells from
<cells>	Cells to remove

Categories

[Floorplan](), [XDC]()

Description

Removes the specified logic instances from a Pblock. Cells are added to a Pblock with the `add_cells_to_pblock` command.

Note: Cells that have been placed will not be unplaced as they are removed from a Pblock. Any current LOC assignments are left intact.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<pblock> - (Required) The name of the Pblock from which to remove the specified instances.

`<cells>` - (Required) One or more cell objects to remove from the specified Pblock.

Send Feedback

Examples

The following example removes the specified cells from the pb_cpuEngine Pblock:

```
remove_cells_from_pblock pb_cpuEngine [get_cells cpuEngine/cpu_dwb_dat_o/*]
```

See Also

- add_cells_to_pblock

remove_cluster_configurations

Removes list of cluster configurations

Syntax

```
remove_cluster_configurations [-quiet] [-verbose] <cluster_configurations>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<cluster_configurations>	List of cluster configurations

Categories

cluster_configuration

Description

Removes a list of cluster configurations. The list of cluster configurations to be removed is specified as input. This command can only remove user defined cluster configurations and cannot remove Vivado default cluster configurations. To identify if a cluster configuration is default or user-defined, use the following snippet:

```
# if it returns 1 cluster configuration is Vivado default otherwise user
defined.
get_property IS_DEFAULT [get_cluster_configurations <cluster configuration
name>]
```

Currently, Vivado supports following clusters management tools:

- Load Sharing Facility (LSF)

- Sun Grid Engine (SGE)

- Simple Linux Utility For Resource Management (SLURM)

Arguments

<cluster_configurations> - (Required) List of cluster configurations object to be removed. You can get a list of existing cluster configurations with the get_cluster_configurations command and provide it as input to the remove_cluster_configurations command.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Send Feedback

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example removes the cluster configuration with name *lsf_medium*.

```
remove_cluster_configurations [get_cluster_configurations lsf_medium]
```

The following example removes all the user defined cluster configurations.

```
remove_cluster_configurations [get_cluster_configurations -filter {IS_DEFAULT == 0}]
```

See Also

- create_cluster_configuration
- get_cluster_configurations
- validate_cluster_configurations

remove_conditions

Remove conditions from a simulation. The names can be specified as Tcl glob pattern

Syntax

```
remove_conditions [-all] [-quiet] [-verbose] [<ConditionObjs>]
```

Usage

Name	Description
[-all]	Remove all conditions
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<ConditionObjs>]	ConditionObjs, id's or names

Categories

Simulation

Description

Remove specified conditions from the current simulation. You must have an open simulation to use this command.

Conditions can be defined prior to starting the simulation. When a condition is added, the simulator evaluates the condition expression anytime a signal change is detected. When a specified condition expression becomes TRUE, the condition commands are run.

The conditions in the current simulation can be reported using the `report_conditions` command.

This command returns nothing, or an error if the command fails.

Arguments

`-all` - (Optional) Remove all conditions in the current simulation.

> **IMPORTANT!** *This option will remove ALL conditions without warning, even if other options are specified.*

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<ConditionObjs> - Specifies one or more condition identifiers to remove from the current simulation. The condition identifiers are returned by the `add_condition` command when the condition is defined.

Examples

The following example removes the specified condition from the current simulation:

```
remove_conditions condition3
```

See Also

- add_condition
- report_conditions

remove_drc_checks

Remove DRC rule check objects from a user rule deck

Syntax

```
remove_drc_checks [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
    -ruledeck <arg> [-quiet] [-verbose] [<patterns>]
```

Returns

Drc_check

Usage

Name	Description
[-of_objects]	Get 'rule_check' objects of these types: 'drc_ruledeck'.
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
-ruledeck	DRC rule deck to modify
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<patterns>]	Match the 'rule_check' objects against patterns. Default: *

Categories

DRC, Object

Description

Remove the specified design rule checks from a drc_ruledeck object.

A rule deck is a collection of design rule checks grouped for convenience, to be run with the `report_drc` command at different stages of the FPGA design flow, such as during I/O planning or placement. The tool comes with a set of factory defined rule decks, but you can also create new user-defined rule decks with the `create_drc_ruledeck` command.

Checks are added to a rule deck using the `add_drc_checks` command.

The DRC rule check object features the `IS_ENABLED` property that can be set to true or false using the `set_property` command. When a new rule check is created, the `IS_ENABLED` property is set to true as a default. Set the `IS_ENABLED` property to false to disable the rule check from being used by `report_drc` without having to remove the rule from the rule deck.

> 💡 **TIP:** *Use the* `reset_drc_check` *command to restore the DRC rule, and its properties, to the default settings.*

This command returns the list of design rule checks that were removed from the specified rule deck.

Arguments

`-of_objects <arg>` - (Optional) Remove the rule checks of the specified drc_ruledeck object from the specified rule deck. This has the effect of removing all the rules in one rule deck from the target rule deck.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Both search <patterns> and `-filter` expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-nocase` - (Optional) Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of `-regexp` only.

`-filter <args>` - (Optional) Filter the results list with the specified expression. The `-filter` argument filters the list of objects returned by the search pattern, based on specified property values. You can find the properties on an object with the `report_property` or `list_property` commands.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note: The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

Chapter 3: Tcl Commands Listed Alphabetically

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input pins that do NOT contain the "RESET" substring within their name:

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (`bool`) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-ruledeck <arg>` - (Required) The name of the rule deck to remove the specified design rule checks from.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<patterns> - (Optional) Remove the design rule checks that match the specified patterns from the rule deck. The default pattern is the wildcard '*' which removes all rule checks from the specified rule deck. More than one pattern can be specified to remove multiple rule checks based on different search criteria.

Note: You must enclose multiple search patterns in braces, {}, or quotes, "", to present the list as a single element.

Examples

The following example removes the rule checks matching the specified filter pattern from the my_rules rule deck:

```
remove_drc_checks -filter {GROUP == AVAL} -ruledeck my_rules
```

The following example disables the specified DRC check without removing it from the rule deck:

```
set_property IS_ENABLED FALSE [get_drc_checks RAMW-1]
```

The following example removes all rule checks from the specified rule deck:

```
remove_drc_checks -ruledeck my_rules
```

UG835 (v2021.1) June 16, 2021
Tcl Command Reference Guide

Send Feedback

www.xilinx.com
1251

See Also

- add_drc_checks
- create_drc_check
- create_drc_ruledeck
- get_drc_checks
- get_drc_ruledecks
- list_property
- report_drc
- report_property
- reset_drc_check

remove_files

Remove files or directories from a fileset

Syntax

```
remove_files [-fileset <arg>] [-quiet] [-verbose] <files>...
```

Returns

List of files that were removed.

Usage

Name	Description
`[-fileset]`	Fileset name
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<files>`	Name of the file(s) to be removed

Categories

[Project](), [Simulation]()

Description

Removes the specified file objects from the current or specified fileset. The file is removed from the current project, but is not removed from the disk.

Files can be specified as file name strings, or as file objects returned by the `get_files` command. When specified as strings, the file is looked for in the current or specified fileset. When the file object is specified by `get_files`, the fileset is defined by the object, and `-fileset` is ignored.

When successful, this command returns nothing. If the specified file is not found, an error is returned.

Arguments

`-fileset <arg>` - (Optional) The name of the fileset to locate the specified files. As a default, the files will be removed from the current fileset as defined by the `current_fileset` command.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Send Feedback

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<files>` - (Required) The name of the files to remove from the project.

Note: If no files are specified, no files are removed.

Examples

The following example removes the file named `C:/Design/top.xdc` from the constraint set `constrs_1`:

```
remove_files -fileset constrs_1 C:/Design/top.xdc
```

Multiple files can be specified as follows:

```
remove_files -fileset sim_1 top_tb1.vhdl top_tb2.vhdl
```

The following example gets all the file objects in the current project, and removes them:

```
remove_files [get_files]
```

> ⚠ **CAUTION!** *This will remove ALL files from your design.*

See Also

- add_files
- current_fileset
- get_files

Send Feedback

remove_forces

Release force on signal, wire, or reg applied using 'add_force' command

Syntax

```
remove_forces [-all] [-quiet] [-verbose] [<ForceObj>...]
```

Usage

Name	Description
`[-all]`	Remove all forces
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`[<ForceObj>]`	ForceObj or id's

Categories

Simulation

Description

Remove the specified force objects, or force IDs from the current simulation.

Forces are applied to specific HDL objects using the `add_forces` command. This command removes those forces from the current simulation.

> ⭐ **IMPORTANT!** *If there are* `force/release` *statements on an HDL object in the test bench or module, these statements are overridden by the* `add_force` *command. When the* `remove_force` *command releases these objects to resume their normal operation, the Verilog* `force/release` *statements resume their effect.*

This command returns nothing if successful, or returns an error if it fails.

Arguments

`-all` - (Optional) Remove all forces from the current simulation.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<ForceObj> - (Optional) Remove only the specified force object or objects. The force ID is returned by the add_force command when the force is created.

Examples

The following example creates a force object using the add_force command, and captures the force ID in a Tcl variable, then removes that force object:

```
set f10 [ add_force reset 1 300 ]
remove_forces $f10
```

The following example removes all force objects from the current simulation:

```
remove_forces -all
```

See Also

- get_objects
- add_force

remove_from_power_rail

Remove power sources from Power Rail

Syntax

```
remove_from_power_rail [-power_sources <args>] [-quiet] [-verbose]
    <power_rail>
```

Usage

Name	Description
[-power_sources]	List of power sources to remove. Can be power rails and/or power supplies
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<power_rail>	Power rail to remove power sources from

Categories

XDC, Floorplan

Send Feedback

remove_gui_custom_command_args

Remove one or more custom command arguments

Syntax

```
remove_gui_custom_command_args -command_name <arg> [-quiet] [-verbose]
    <names>...
```

Usage

Name	Description
-command_name	name of custom command whose arguments are being removed.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<names>	name of one or more custom command arguments to remove.

Categories

GUIControl

Description

Remove one or more GUI custom command arguments of a particular GUI custom command.

You can use `get_gui_custom_commands` to determine the list of defined custom commands. You can use `get_gui_custom_command_args` to determine the list of defined GUI custom command arguments for a particular GUI custom command.

Arguments

`-command_name` - (REquired) SPecifies the name of GUI custom command whose arguments are being removed.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<names> - (Required) One or more argument names to remove.

Examples

Remove GUI custom command arguments with name 'arg1' and 'arg2' of a GUI custom command with name 'cmd_1' :

```
remove_gui_custom_command_args -command_name cmd_1 {arg1 arg2}
```

See Also

- create_gui_custom_command
- create_gui_custom_command_arg
- get_gui_custom_commands
- get_gui_custom_command_args
- remove_gui_custom_commands

remove_gui_custom_commands

Remove one or more custom commands

Syntax

```
remove_gui_custom_commands [-quiet] [-verbose] <names>...
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<names>	name of one or more custom commands to remove

Categories

GUIControl

Description

Remove one or more GUI custom commands.

You can use `get_gui_custom_commands` to determine the list of defined custom commands.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<names> - (Required) One or more command names to remove.

Examples

The following example removes the GUI custom commands with names 'abc' and 'xyz':

```
remove_gui_custom_commands {abc xyz}
```

Send Feedback

See Also

- create_gui_custom_command
- create_gui_custom_command_arg
- get_gui_custom_command_args
- get_gui_custom_commands
- remove_gui_custom_command_args

remove_hw_hbm_pc

De-select Pseudo Channel from Activity Monitor on the specified hardware HBM(s). Must specify the Memory Controller number first, follow by Pseudo Channel number.

Syntax

```
remove_hw_hbm_pc [-quiet] [-verbose] <mc_num> <pc_num> <hw_objects>
```

Usage

Name	Description
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<mc_num>`	Memory Controller number to de-select: 0 to 7
`<pc_num>`	Pseudo Channel number to de-select: 0 or 1
`<hw_objects>`	hardware objects

Categories

Hardware

Description

The `remove_hw_hbm_pc` command removes a memory channel (mc)/pseudo channel (pc) that has been previously added to an HBM activity monitor using the `add_hw_hbm_pc` command.

> 💡 **TIP:** *The HBM activity monitor must not be running when adding or removing pseudo channels.*

This command returns nothing if successful, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Send Feedback

<mc_num> - (Required) Specifies one of the 8 memory channels of the HBM controller, numbered 0 to 7.

<pc_num> - (Required) Specifies one of the 2 pseudo channels, numbered 0 or 1.

<hw_objects> - (Required) Specifies the hw_hbms to remove the pseudo channel from.

Examples

The following example removes the previously added pseudo channel from the HBM activity monitor, and then runs the activity monitor:

```
remove_hw_hbm_pc 2 0 [get_hw_hbms *HBM_2]
run_hw_hbm_amon [get_hw_hbms *HBM_2]
```

See Also

- commit_hw_hbm
- current_hw_device
- current_hw_target
- get_hw_hbms
- pause_hw_hbm_amon
- refresh_hw_hbm
- resume_hw_hbm_amon
- run_hw_hbm_amon
- stop_hw_hbm_amon

remove_hw_probe_enum

Remove enumerated name-value pairs from a hw_probe enumeration.

Syntax

```
remove_hw_probe_enum [-no_gui_update] [-list <args>] [-remove_all] [-quiet]
    [-verbose] <hw_probe>
```

Usage

Name	Description
[-no_gui_update]	Defer GUI update.
[-list]	List of enumerated names to remove.
[-remove_all]	Remove the whole enumeration for a hardware probe. Default: 0
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hw_probe>	ILA hardware probe object.

Categories

Hardware

Description

Remove the enumerated name/value pairs defined on a specified hw_probe object.

The enumerated names (ENUM property) are added to a hw_probe object using the `add_hw_probe_enum` command. This command removes those defined properties.

This command returns nothing if successful, or returns an error if it fails.

Arguments

`-no_gui_update` - (Optional) Do not update the GUI in the Vivado logic analyzer to remove the enumerated values of the probe.

`-list` <args> - (Optional) Remove the specified list of enumerated names from the specified <hw_probe> object.

> 💡 **TIP:** *The list of names can be specified as a list object, or as a simple list of names.*

`-remove_all` - (Optional) Remove all of the ENUM properties defined on the specified <hw_probe> object. This option cannot be used with the `-list` option.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_probe> - (Required) Specify the hw_probe objects to remove the ENUM property from.

Examples

The following example removes the list of enumerated names from the specified hw_probe object:

```
remove_hw_probe_enum -list {WHITE YELLOW GREY} \
[get_hw_probes op1 -of_objects [current_hw_ila]]
```

See Also

- add_hw_probe_enum
- current_hw_device
- current_hw_ila
- get_hw_devices
- get_hw_ilas
- get_hw_probes
- get_hw_vios
- report_property

remove_hw_sio_link

Remove an existing hardware SIO link.

Syntax

```
remove_hw_sio_link [-quiet] [-verbose] <hw_sio_links>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hw_sio_links>	hardware SIO links

Categories

Hardware

Description

Removes the specified communication links between TX and RX objects on the GTs of the IBERT debug core defined on the current hardware device.

Vivado Serial I/O analyzer is a link-based analyzer, which lets you link between any transmitter and receiver within the IBERT design. The links define the communication paths and protocols between transmitters and receivers of the GigaBit transceivers on the device. This command removes those links.

This command returns a list of link objects on the IBERT debug core, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_sio_links> - (Required) Specify one or more hw_sio_link objects to remove. The hw_sio_link must be specified as an object as returned by the `create_hw_sio_link` or `get_hw_sio_links` commands.

Example

The following example removes the specified communication link on the IBERT debug core:

```
remove_hw_sio_link [get_hw_sio_links -filter {DESCRIPTION == "Link2"}]
```

See Also

- create_hw_sio_link
- create_hw_sio_linkgroup
- current_hw_device
- get_hw_sio_iberts
- get_hw_sio_links
- get_hw_sio_linkgroups
- remove_hw_sio_linkgroup

remove_hw_sio_linkgroup

Remove an existing hardware SIO link group.

Syntax

```
remove_hw_sio_linkgroup [-quiet] [-verbose] <hw_sio_linkgroups>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hw_sio_linkgroups>	hardware SIO linkgroups

Categories

Hardware

Description

Removes the specified group that associates communication links between TX and RX objects on the GTs of the IBERT debug core defined on the current hardware device.

Vivado Serial I/O analyzer is a link-based analyzer. The links define the communication paths and protocols between transmitters and receivers of the GigaBit transceivers on the device. Link groups, or hw_sio_linkgroup objects, let you associate links into related groups, to collectively configure properties and run scans.

> **TIP:** *The* `remove_hw_sio_linkgroup` *command removes the specified association, but does not remove the underlying communication links. Us the* `remove_hw_sio_link` *command to remove those objects.*

This command returns nothing if successful, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Send Feedback

Note: Message limits can be defined with the `set_msg_config` command.

<hw_sio_linkgroups> - (Required) Specify one or more hw_sio_linkgroup objects to remove. The hw_sio_linkgroup must be specified as an object as returned by the `create_hw_sio_linkgroup` or `get_hw_sio_linkgroups` commands.

Example

The following example removes the specified linkgroup:

```
remove_hw_sio_linkgroup [get_hw_sio_linkgroups {LINKGROUP_0}]
```

See Also

- create_hw_sio_link
- create_hw_sio_linkgroup
- current_hw_device
- get_hw_sio_iberts
- get_hw_sio_links
- get_hw_sio_linkgroups
- remove_hw_sio_link

Send Feedback

remove_hw_sio_scan

Remove an existing hardware SIO scan.

Syntax

```
remove_hw_sio_scan [-quiet] [-verbose] <hw_sio_scans>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hw_sio_scans>	hardware SIO scans

Categories

[Hardware](#)

Description

Remove the specified serial I/O analyzer scan object.

This command returns nothing if successful, or returns an error if the command fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_sio_scans> - (Required) Specify one or more hw_sio_scan objects to remove. The hw_sio_scan must be specified as an object as returned by the `create_hw_sio_scan` or `get_hw_sio_scans` commands.

Send Feedback

Example

The following example removes the specified SIO scan:

```
remove_hw_sio_scan [get_hw_sio_scans {SCAN_2}]
```

See Also

- create_hw_sio_scan
- get_hw_sio_scans
- run_hw_sio_scan
- stop_hw_sio_scan
- wait_on_hw_sio_scan
- write_hw_sio_scan

Send Feedback

remove_hw_sio_sweep

Remove an existing hardware SIO sweep.

Syntax

```
remove_hw_sio_sweep [-quiet] [-verbose] <hw_sio_sweeps>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hw_sio_sweeps>	hardware SIO sweeps

Categories

Hardware

Description

Remove the specified serial I/O analyzer sweep scan object.

This command returns nothing if successful, or returns an error if the command fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_sio_sweeps> - (Required) Specify one or more hw_sio_sweep objects to remove. The hw_sio_sweep must be specified as an object as returned by the `create_hw_sio_sweep` or `get_hw_sio_sweeps` commands.

Example

The following example removes the specified sweep scan object:

```
remove_hw_sio_sweep [get_hw_sio_sweeps {SWEEP_3}]
```

See Also

- create_hw_sio_sweep
- get_hw_sio_sweeps
- run_hw_sio_sweep
- stop_hw_sio_sweep
- wait_on_hw_sio_sweep
- write_hw_sio_sweep

remove_net

Remove nets from the current design

Syntax

```
remove_net [-prune] [-quiet] [-verbose] <nets>...
```

Usage

Name	Description
[-prune]	When performing net removal, remove pins and ports which are left unconnected as a result of the remove_net operation.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<nets>	List of nets to remove

Categories

Netlist

Description

Remove the specified net from the netlist of an open Synthesized or Implemented Design.

Note: You cannot remove nets from library macros, also called macro-primitives.

To remove a bus, you must specify the primary bus name, and not specify a bus index. This ensures that the entire bus is removed, and not just a portion of the bits associated with the bus. You can resize a bus, eliminating bits of the bus, using the `resize_net_bus` command.

Netlist editing changes the in-memory view of the netlist in the current design. It does not change the files in the source fileset, or change the persistent design on the disk. Changes made to the netlist may be saved to a design checkpoint using the `write_checkpoint` command, or may be exported to a netlist file such as Verilog, VHDL, or EDIF, using the appropriate `write_*` command.

Note: Netlist editing is not allowed on the elaborated RTL design.

Arguments

`-prune` - (Optional) Prune, or remove, any unconnected hierarchical pins, ports, or nets, as a result of removing the specified net.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<nets>` - (Required) The list of nets to remove from the netlist of the current design.

Example

Using the following connection network:

```
leaf_cell1/pin1 > net1 > block1/pin1 >
   topnet
< block2/pin1 < net2 < leaf_cell2/pin1
```

This example will remove block1/pin1, block2/pin1, net1, and net2, but will not prune the pins on the leaf cells:

```
remove_net topnet -prune
```

The following example illustrates the warning returned when trying to remove one bit of a bus net, and then removes the entire bus by specifying the root name:

```
remove_net DataIn_pad_1_i[0]
WARNING: [Coretcl-82] No nets matched 'DataIn_pad_1_i[0]'.
remove_net DataIn_pad_1_i
```

See Also

- create_net
- disconnect_net
- resize_net_bus
- write_checkpoint
- write_edif
- write_verilog
- write_vhdl

Send Feedback

remove_pin

Remove pins from the current design

Syntax

```
remove_pin [-quiet] [-verbose] <pins>...
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<pins>	List of pins to remove

Categories

Netlist

Description

Remove pins from the current netlist in either an open Synthesized or Implemented design.

Note: You cannot remove pins from library macros, or macro-primitives.

To remove a bus pin, you must specify the primary pin name, and not specify a bus index. This ensures that the entire bus pin is removed, and not just a portion of the bits associated with the bus. You can resize a bus pin, eliminating bits, using the `resize_pin_bus` command.

Netlist editing changes the in-memory view of the netlist in the current design. It does not change the files in the source fileset, or change the persistent design on the disk. Changes made to the netlist may be saved to a design checkpoint using the `write_checkpoint` command, or may be exported to a netlist file such as Verilog, VHDL, or EDIF, using the appropriate `write_*` command.

Note: Netlist editing is not allowed on the elaborated RTL design.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<pins> - (Required) List of pins to remove from the netlist. The pins must be specified hierarchically by the cell instance the pin is found on.

Examples

The following example removes the specified pin from the cpuEngine in the in-memory netlist of the current design:

```
remove_pin cpuEngine/inPin
```

See Also

• create_pin

Send Feedback

remove_port

Remove the given list of top ports from the netlist.

Syntax

```
remove_port [-quiet] [-verbose] <ports>...
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<ports>	Ports and/or bus ports to remove

Categories

PinPlanning

Description

Removes the specified ports or buses.

To remove a bus port, you must specify the primary port name, and not specify a bus index. This ensures that the entire bus port is removed, and not just a portion of the bits associated with the bus. You can resize a bus port, eliminating bits, using the `resize_port_bus` command.

The `remove_port` command will remove ports that have been added with the `create_port` command, but cannot delete ports that are defined in the RTL or netlist design.

Netlist editing changes the in-memory view of the netlist in the current design. It does not change the files in the source fileset, or change the persistent design on the disk. Changes made to the netlist may be saved to a design checkpoint using the `write_checkpoint` command, or may be exported to a netlist file such as Verilog, VHDL, or EDIF, using the appropriate `write_*` command.

Note: Netlist editing is not allowed on the elaborated RTL design.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

Send Feedback

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<ports> - One or more names of ports to remove.

Examples

The following example deletes the specified port:

```
remove_port PORT0
```

The following example deletes the two specified ports of a bus:

```
remove_port BUS[1] BUS[2]
```

The following example deletes both the N and P sides of a differential pair port:

```
remove_port D_BUS_P[0]
```

Note: Deleting either the N or the P side of a differential pair will also delete the other side of the pair.

See Also

- create_cell
- create_net
- create_pin
- create_port
- create_interface
- place_ports
- resize_port_bus
- write_checkpoint
- write_edif
- write_verilog
- write_vhdl

remove_wave

Removes wave objects from the current wave configuration

Syntax

```
remove_wave [-of <args>] [-quiet] [-verbose] <items>...
```

Usage

Name	Description
[-of]	the wave configuration, group, or virtual bus to search Default: the current wave configuration
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<items>	wave objects to remove

Categories

Waveform

rename_cell

rename a cell

Syntax

```
rename_cell -to <arg> [-quiet] [-verbose] <cell>...
```

Usage

Name	Description
`-to`	New name
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<cell>`	Cell to rename

Categories

Netlist

Description

Rename a single hierarchical or leaf-level cell in the current synthesized or implemented design.

> **TIP:** *You cannot rename cells with DONT_TOUCH property set to TRUE.*

Netlist editing changes the in-memory view of the netlist in the current design. It does not change the files in the source fileset, or change the persistent design on the disk. Changes made to the netlist may be saved to a design checkpoint using the `write_checkpoint` command, or may be exported to a netlist file such as Verilog, VHDL, or EDIF, using the appropriate `write_*` command.

Note: Netlist editing is not allowed on the elaborated RTL design.

Changes to the names of cells, nets, pins, and ports, will also affect the design constraints defined in the in-memory design. Constraints are automatically modified to target the new object name, however these are not written back to the source XDC file. Saving the modified in-memory design using `write_checkpoint` will save both the renamed objects and modified constraints.

This command returns nothing if successful, or an error if it fails.

Arguments

`-to <arg>` - (Required) Specify the new name to assign to the specified cell. Specified names can not contain Tcl special characters: `'"\{};$#`

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<cell>` - (Required) Instance name of the cell to rename.

Examples

The following example changes the name of the hierarchical or1200_cpu cell as specified:

```
rename_cell -to or1200_gpu or1200_cpu
```

See Also

- connect_net
- create_cell
- create_net
- remove_cell
- set_hierarchy_separator
- write_checkpoint
- write_edif
- write_verilog
- write_vhdl

Send Feedback

rename_net

rename a net

Syntax

```
rename_net -to <arg> [-quiet] [-verbose] <net>...
```

Usage

Name	Description
-to	New name
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<net>	Net to rename

Categories

[Netlist](#)

Description

Rename a net in the current synthesized or implemented design.

The following are limitations with regard to renaming nets:

- You cannot rename nets that have DONT_TOUCH or MARK_DEBUG properties set to TRUE.

- You cannot rename individual bits of a bus net, but you can collectively rename the whole bus.

Netlist editing changes the in-memory view of the netlist in the current design. It does not change the files in the source fileset, or change the persistent design on the disk. Changes made to the netlist may be saved to a design checkpoint using the `write_checkpoint` command, or may be exported to a netlist file such as Verilog, VHDL, or EDIF, using the appropriate `write_*` command.

Note: Netlist editing is not allowed on the elaborated RTL design.

Changes to the names of cells, nets, pins, and ports, will also affect the design constraints defined in the in-memory design. Constraints are automatically modified to target the new object name, however these are not written back to the source XDC file. Saving the modified in-memory design using `write_checkpoint` will save both the renamed objects and modified constraints.

This command returns nothing if successful, or an error if it fails.

Send Feedback

Arguments

`-to <arg>` - (Required) Specify the new name to assign to the specified net. Specified names can not contain Tcl special characters: `'"\{};$#`

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<net>` - (Required) Name of a net to rename.

Examples

The following example renames the specified bus signal:

```
rename_net -to dataOut dout
```

See Also

- connect_net
- create_cell
- create_net
- create_pin
- create_port
- remove_cell
- set_hierarchy_separator
- write_checkpoint
- write_edif
- write_verilog
- write_vhdl

Send Feedback

rename_pin

rename a pin

Syntax

```
rename_pin -to <arg> [-quiet] [-verbose] <pin>...
```

Usage

Name	Description
-to	New name
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<pin>	Pin to rename

Categories

Netlist

Description

Rename the specified pin on a hierarchical cell in the current synthesized or implemented design.

The following are limitations with regard to renaming pins:

- Pins on primitive cells cannot be renamed.

- A pin on a hierarchical cell that has the DONT_TOUCH property can be renamed, but a pin on an hierarchical cell inside a DON'T_TOUCH cell cannot be renamed.

- You cannot rename individual bits of a bus pin, but you can collectively rename the whole bus.

Netlist editing changes the in-memory view of the netlist in the current design. It does not change the files in the source fileset, or change the persistent design on the disk. Changes made to the netlist may be saved to a design checkpoint using the `write_checkpoint` command, or may be exported to a netlist file such as Verilog, VHDL, or EDIF, using the appropriate `write_*` command.

Note: Netlist editing is not allowed on the elaborated RTL design.

Changes to the names of cells, nets, pins, and ports, will also affect the design constraints defined in the in-memory design. Constraints are automatically modified to target the new object name, however these are not written back to the source XDC file. Saving the modified in-memory design using `write_checkpoint` will save both the renamed objects and modified constraints.

This command returns nothing if successful, or an error if it fails.

Arguments

`-to <arg>` - (Required) The new name to assign to the specified pin. The new name only needs to specify the pin name, rather than the whole hierarchical name of the pin. Specified names can not contain Tcl special characters: `' " \ { } ; $ #`

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<pin> - (Required) The hierarchical name of a pin, starting with the instance name of the cell it is found on.

Examples

The following example renames the specified pin:

```
rename_pin -to in1 egressLoop[0].egressFifo/I1
```

The following example shows the error that is returned when you try to rename a single bit of a bus, and then renames the whole bus pin:

```
rename_pin -to din[0] egressLoop[0].egressFifo/buffer_fifo/dataInput[0]
WARNING: [Coretcl 2-1480] rename_pin can not rename bits of a bus, \
use resize_pin_bus instead.
rename_pin -to dataInput egressLoop[0].egressFifo/buffer_fifo/din
```

See Also

- connect_net
- create_cell
- create_net
- create_pin
- create_port
- remove_pin
- set_hierarchy_separator
- write_checkpoint
- write_edif

Send Feedback

- write_verilog
- write_vhdl

rename_port

rename a port

Syntax

```
rename_port -to <arg> [-quiet] [-verbose] <port>...
```

Usage

Name	Description
-to	New name
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<port>	Port to rename

Categories

Netlist

Description

Rename a single port in the current synthesized or implemented design.

> 💡 **TIP:** *You cannot rename individual bits of a bus port, but you can collectively rename the whole bus.*

Netlist editing changes the in-memory view of the netlist in the current design. It does not change the files in the source fileset, or change the persistent design on the disk. Changes made to the netlist may be saved to a design checkpoint using the `write_checkpoint` command, or may be exported to a netlist file such as Verilog, VHDL, or EDIF, using the appropriate `write_*` command.

Note: Netlist editing is not allowed on the elaborated RTL design.

Changes to the names of cells, nets, pins, and ports, will also affect the design constraints defined in the in-memory design. Constraints are automatically modified to target the new object name, however these are not written back to the source XDC file. Saving the modified in-memory design using `write_checkpoint` will save both the renamed objects and modified constraints.

This command returns nothing if successful, or an error if it fails.

Arguments

`-to <arg>` - (Required) Specify the new name to assign to the specified port. Specified names can not contain Tcl special characters: `'"\{};$#`

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<port>` - (Required) Name of the port to rename.

Examples

The following example renames the specified bus port:

```
rename_port -to wbInputData wbInDat
```

See Also

- connect_net
- create_net
- create_port
- remove_port
- set_hierarchy_separator
- write_checkpoint
- write_edif
- write_verilog
- write_vhdl

rename_ref

rename a cell ref

Syntax

```
rename_ref [-ref <arg>] [-to <arg>] [-prefix_all <arg>] [-quiet] [-verbose]
```

Usage

Name	Description
[-ref]	Cell ref to rename
[-to]	New name
[-prefix_all]	Rename all eligible hierarchical cell refs in the current design. Construct the new name using the given prefix plus the original name
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Netlist

Description

Rename the reference name of a single non-primitive cell, or apply a reference prefix to all non-primitive cells in the current synthesized or implemented design.

This command provides a mechanism to change the non-primitive reference names in the current design so that they do not collide with the reference names in another design. This lets two modules or designs be synthesized or simulated together, while avoiding any name collisions between the two designs.

This command returns nothing when renaming the reference a single cell, and returns the number of cells renamed when used with `-prefix_all`. If the command fails, an error is returned.

Arguments

`-ref <arg>` - (Optional) Specify the current reference name of a non-primitive cell.

`-to <arg>` - (Optional) Change the reference name to the specified value.

`-prefix_all <arg>` - (Optional) Apply the specified prefix to the reference names of all non-primitive cells in the current design, except the top module.

Send Feedback

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example changes the specified reference name to the value indicated:

```
rename_ref -ref usbf_top -to MOD1_usbf_top
```

The following example applies the specified reference name prefix to all non-primitive cells in the current design:

```
rename_ref -prefix_all MOD1_
```

See Also

- launch_simulation
- synth_design
- write_edif
- write_verilog
- write_vhdl

reorder_files

Change the order of source files in the active fileset

Syntax

```
reorder_files [-fileset <arg>] [-before <arg>] [-after <arg>] [-front]
    [-back] [-auto] [-disable_unused] [-quiet] [-verbose] <files>...
```

Usage

Name	Description
[-fileset]	Fileset to reorder
[-before]	Move the listed files before this file
[-after]	Move the listed files after this file
[-front]	Move the listed files to the front (default)
[-back]	Move the listed files to the back
[-auto]	Automatically re-orders the given fileset
[-disable_unused]	Disables all files not associated with the TOP design unit
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<files>	Files to move

Categories

Project

Description

Reorders source files in the specified fileset. Takes the files indicated and places them at the front of, the back of, or before or after other files within the fileset. This command also has an auto reorder feature that reorders the files based on the requirements of the current top module in the design.

Arguments

`-fileset <arg>` - (Optional) The fileset in which to reorder files. The default is the sources_1 source fileset.

`-before <arg>` - (Optional) Place the specified files before this file in the fileset. The file must be specified with the full path name in the fileset.

`-after <arg>` - (Optional) Place the specified files after this file in the fileset. The file must be specified with the full path name in the fileset.

Send Feedback

`-front` - (Optional) Place the specified files at the front of the list of files in the fileset.

`-back` - (Optional) Place the specified files at the back of the list of files in the fileset.

`-auto` - (Optional) Enable automatic reordering based on the hierarchy requirements of the current top-module in the project. Often used after changing the top module with the "`set_property top`" command.

`-disable_unused` - (Optional) Disable any files not currently used by the hierarchy based on the top-module. Often used after changing the top module with the "`set_property top`" command.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<files>` - (Required) One or more files to relocate in the fileset. Files must be specified by their full path name in the fileset, and are reordered in the order they are specified.

Examples

The following example takes the specified files and moves them to the front of the source fileset:

```
reorder_files -front {C:/Data/FPGA/file1.vhdl C:/Data/FPGA/file2.vhdl}
```

Note: The default source fileset is used in the preceding example since the `-fileset` argument is not specified.

The following example sets a new top_module in the design, and then automatically reorders and disables unused files based on the hierarchy of the new top-module:

```
set_property top block1 [current_fileset]
reorder_files -auto -disable_unused
```

See Also

- add_files
- create_fileset
- current_fileset
- remove_files

replace_bd_cell

Replace cell1 with cell2 by disconnecting connections to cell1 and connecting those connections to cell2.

Syntax

```
replace_bd_cell [-preserve_name] [-preserve_configuration] [-quiet]
    [-verbose] [<cell1>] [<cell2>...]
```

Returns

TCL_OK, TCL_ERROR if failed.

Usage

Name	Description
[-preserve_name]	cell2 will rename as cell1's name, cell1 rename as cell1name_old
[-preserve_configuration]	preserve configuration of cell1 on cell2
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<cell1>]	Cell with connections that are to be disconnected.
[<cell2>]	Cell to be connected to connections that were disconnected from cell1.

Categories

IPIntegrator

Description

Move the connections currently assigned to one IP integrator cell to another IP integrator cell in the current design. This is intended to help you quickly replace one cell with another by moving connections from the source cell to the target cell.

The current, or existing cell, will be relocated from its current position in the block design, and the new replacing cell will be placed at that location. Connections to the pins and interface pins on the cell are preserved where possible, and result in a Critical Warning when connections must be removed.

IMPORTANT! *This command is not supported by the UNDO command.*

This command returns TCL_OK if it is successful, or returns TCL_ERROR if it fails.

Arguments

`-preserve_name` - (Optional) Apply the name of the current cell to the new cell that is replacing it. The existing cell will be renamed as `<instance>_old`.

`-preserve_configuration` - (Optional) Apply the configuration settings of the current cell to the new cell that is replacing it.

Note: Any configuration options on the original cell that cannot be applied to the new cell will be reported as a warning, and then skipped.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<cell1> - (Required) The IP integrator cell to remove connections from and to replace with a new cell. The cell can be specified either by name, or as a bd_cell object returned by the `get_bd_cells` command.

Note: The cell is not removed from the IP integrator subsystem design, but is relocated to make room for the new cell.

<cell2> - The IP integrator cell that replaces the existing cell (<cell1>) and that existing connections are applied to. The cell can be specified either by name, or as a bd_cell object returned by the `get_bd_cells` command.

Example

The following example moves the connections from the specified cell, lmb_v10_1, to pins and interface pins of the same name on another cell, lmb_bram_cntlr_1:

```
replace_bd_cell [get_bd_cells /lmb_v10_1] [get_bd_cells \
    /lmb_bram_if_cntlr_1]
CRITICAL WARNING: [BD 41-1164] The interface pin 'LMB_Sl_0' with
bus definition 'xilinx.com:interface:lmb:1.0' is not found on the
cell '/lmb_bram_if_cntlr_1'. Its connection to the interface
net 'Conn' has been removed.
CRITICAL WARNING: [BD 41-1166] The pin 'SYS_Rst' is not found
on the cell '/lmb_bram_if_cntlr_1'. Its connection to the net
'sys_rst_1' has been removed.
0
```

Note: Critical Warnings are returned when pin mismatches necessitate the removal of existing connections.

See Also

- create_bd_cell
- get_bd_cells

report_bd_diffs

Report differences between two block designs. Note this TCL command may compare bus-interface parameters from IP .xit files that are not reported in the stand-alone diffbd executable.

Syntax

```
report_bd_diffs [-format <arg>] [-file <arg>] [-open_html] [-brief]
    [-strict] [-fast] [-return_string] [-depth <arg>] [-crossprobe]
    [-repository <arg>] [-take_snapshot] [-diff_snapshot] [-quiet]
    [-verbose] <design1> <design2>
```

Returns

0 if no functional differences, >0 if different, -1 on error.

Usage

Name	Description
[-format]	Format: html or text Default: text
[-file]	Optional output file name. *.htm* implies HTML format
[-open_html]	Open HTML report in a browser. Sets format to HTML
[-brief]	Output only whether files differ. Don't write report
[-strict]	Non-functional changes are treated as functional changes
[-fast]	Read BD JSON directly instead of creating in-memory BDs when possible. Will not compare Bus-Interface parameters from .XIT files (same as stand-alone diffbd)
[-return_string]	(Text only) Return the report as a string
[-depth]	HTML Display Depth for equal items. Used to limit HTML file size. Does not affect not-equal items. Implies HTML format. Default: 4
[-crossprobe]	Enable links on HTML report to select Vivado objects. Implies open_html
[-repository]	User repository for designs on disk
[-take_snapshot]	Take a snapshot of the current block design
[-diff_snapshot]	Compare the current block design against a snapshot
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<design1>	Name or file path of first block design to compare
<design2>	Name or file path of second block design to compare

Categories

IPIntegrator

Description

Analyze and report the difference between two block design files (.bd).

> 💡 **TIP:** *There is also a standalone version of this command, `diffbd`, that can be run from the command line. You can find out more information about this command by typing `diffbd -h` from the command line. Refer to the Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator (UG994) for more informaiton.*

This command performs a non-graphical comparison of two block designs to let you compare revisions of a block design from within revision control systems. Block designs must be specified as BD objects, as returned by `current_bd_design`, or `get_bd_designs` commands. The design objects can have the same name, but be returned from different `.bd` files. An error will be returned if the BD objects refer to the same design.

The differences reported include additions, or changes to the IP in use in the block diagram, changes to design properties or parameters, changes to the design hierarchy, changes to connectivity, and changes memory addressing.

This command returns the difference report for the two block designs specified, or returns an error if it fails.

Arguments

`-format [ text | html ]` - (Optional) Specifies the format of the difference report as either text, or HTML. Unless the `-file` option is specified, text format returns the report to the Tcl console by default, and HTML format writes the report to `diff.html` in the current working directory. The default format is text.

`-file <arg>` - (Optional) Write the difference report into the specified file. Using a file suffix of `.htm` implies an HTML output, unless the `-format` option specifies text.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-open_html` - (Optional) Opens the specified HTML report in the default web browser. This option implies a report format of HTML, and cannot be used with `-format text`.

Note: This option does not support the Internet Explorer or Edge web browsers. You can open the HTML report in these browsers after generation, but the tool will not open them automatically.

`-brief` - (Optional) Returns a brief message indicating whether the two designs differ, but does not detail how they may differ. Can be used as a pre-check for scripting purposes.

Send Feedback

`-strict` - (Optional) Examines and reports all changes as functional changes, returning a value of 0 if there are no changes. Functional changes have to do with the design itself: objects, connections, parameters. Non-functional changes have to do with the metadata of the BD, such as tool version, and do not result in a difference in functionality in the design. By default only functional changes are reported as differences. However, the use of `-strict` reports any changes in the difference report.

`-fast` - (Optional) Perform a quick compare by reading the BD JSON files directly rather than opening in-memory block designs.

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-depth <arg>` - (Optional) Specifies the depth of the design hiearchy to report when the designs are equal, or matching. Note that any differences in the design will always be reported regardless of this option.

`-crossprobe` - (Optional) Add cross-probing links to the HTML report format. This allows items from the report to be selected from an open block design. This implies the HTML format and `-open_html`. This option cannot be used with `-format text`.

`-repository <arg>` - (Optional) Specify an IP repository for block designs that include IP from outside the Xilinx standard IP catalog. The location of the IP repository is needed to resolve any references to non-standard IP used in the design.

`-take_snapshot` - (Optional) Take an in-memory snapshot of the current open block design. This option does not require design names, or additional options.

`-diff_snapshot` - (Optional) Compare the current open block design against a previous in-memory snapshot of the block design. This lets you see what changes have been made to the block design since the snapshot was taken.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<design1> - (Optional) specify a block design to diff with a second block design. The block design can be specified by `current_bd_design`, or `get_bd_designs` Tcl commands; or can be specified by path and file name.

Note: If the path is not specified with the file name, the file will be looked for in the current project, or the current working directory.

<design2> - (Optional) specify a block design to diff with the first block design. The block design can be specified by `current_bd_design`, or `get_bd_designs` Tcl commands; or can be specified by path and file name.

Examples

The following example compares the two specified block designs, one from a file path, the second from the current open block design, and creates and opens an HTML report format with the specified depth:

```
report_bd_diffs C:/Data/Base_Zynq_MPSoC.bd [current_bd_design] \
-file C:/Data/diffs5.htm -open_html -depth 5
```

The following example takes a snapshot of the current open block design:

```
report_bd_diffs -take_snapshot
```

See Also

- current_bd_design
- get_bd_designs

Send Feedback

report_bps

Print details of the given breakpoint objects

Syntax

```
report_bps [-quiet] [-verbose] [<BreakPointObjs>...]
```

Returns

Print the breakpoints id, file_name and line_number to the console in textual format.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<BreakPointObjs>]	List of breakpoint objects to report

Categories

Simulation

Description

Report a specific breakpoint object, or report all breakpoints in the current simulation. You must have an open simulation for this command to return anything.

A breakpoint is a user-determined stopping point in the source code used for debugging the design. When simulating a design with breakpoints, simulation of the design stops at each breakpoint to let you examine values and verify the design behavior.

This command returns the filename and line number of the specified breakpoints, or of all breakpoints in the current simulation, or returns an error if the command fails.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Send Feedback

Note: Message limits can be defined with the `set_msg_config` command.

<BreakPointObjs> - Specifies one or more breakpoint objects in the current simulation to report. The breakpoint object is returned by the `add_bp` command when the breakpoint is added to the simulation.

Examples

The following example reports all breakpoints in the current simulation:

```
report_bps
```

This example reports the specified breakpoints in the current simulation:

```
report_bps bp1 bp2 bp5
```

See Also

- add_bp
- remove_bps

Send Feedback

report_bus_skew

Report timing paths

Syntax

```
report_bus_skew [-delay_type <arg>] [-setup] [-hold] [-no_detailed_paths]
    [-max_paths <arg>] [-nworst <arg>] [-unique_pins] [-path_type <arg>]
    [-sort_by_slack] [-input_pins] [-no_header] [-significant_digits <arg>]
    [-file <arg>] [-append] [-return_string] [-warn_on_violation]
    [-rpx <arg>] [-cells <args>] [-quiet] [-verbose]
```

Usage

Name	Description
[-delay_type]	Type of path delay: Values: max, min, min_max Default: min_max
[-setup]	Report max delay endpoint timing paths (equivalent to -delay_type max)
[-hold]	Report min delay endpoint timing paths (equivalent to -delay_type min)
[-no_detailed_paths]	Only report top level summary table
[-max_paths]	Maximum number of paths to output per bus skew constraint: Value >=1 Default: 1
[-nworst]	List up to N worst paths per endpoint per constraint: Value >=1 Default: 1
[-unique_pins]	For each unique set of pins, show at most 1 path per bus skew constraint
[-path_type]	Format for path report: Values: short, full, full_clock, full_clock_expanded Default: full_clock_expanded
[-sort_by_slack]	Sort summary and per-constraint sections by slack
[-input_pins]	Show input pins in path
[-no_header]	Do not generate a report header
[-significant_digits]	Number of digits to display: Range: 0 to 3 Default: 3
[-file]	Filename to output results to. (send output to console if -file is not used)
[-append]	Append the results to file, don't overwrite the results file
[-return_string]	Return report as string
[-warn_on_violation]	Issue a critical warning when the report contains a timing violation
[-rpx]	Filename to output interactive results to.
[-cells]	run report_bus_skew on the specified hierarchical cell(s)
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Report](#), [Timing](#)

Send Feedback

Description

Report the calculated bus skew among the signals constrained by `set_bus_skew`.

The bus skew requirement applies to both the slow and fast corners. The Vivado tool determines the earliest and the latest arrival among all the signals of the bus and calculates the bus skew for both the Slow and Fast process corner, and reports the worst case skew. Each signal of the bus is reported relative to a reference signal from the same bus. Note that the reference signal can be different for each signal of the bus, which ever results in the worst bus skew for that signal.

The bus skew report can be written to the Tcl console or command shell, assigned to a return string, or saved to a file.

This command returns the bus skew report as specified, or returns an error if it fails.

Arguments

`-delay_type <arg>` - (Optional) Specifies the type of delay to analyze when running the bus skew report. The valid values are min, max, min_max, max_rise, max_fall, min_rise, min_fall. The default setting for `-delay_type` is max.

`-setup` - (Optional) Check for setup violations. This is the same as specifying `-delay_type max`.

`-hold` - (Optional) Check for hold violations. This is the same as specifying `-delay_type min`.

> 💡 **TIP:** `-setup` and `-hold` can be specified together, which is the same as specifying `-delay_type min_max`.

`-no_detailed_paths` - (Optional) By default the bus skew report includes detailed path analysis for the signals reported. This option disables the detailed path in the bus skew report.

`-max_paths <arg>` - (Optional) The maximum number of paths to report per constraint. This is specified as a value greater than or equal to 1. By default the `report_bus_skew` command will report the single worst timing path per bus skew constraint.

`-nworst <arg>` - (Optional) The number of timing paths per endpoint to output in the bus skew report. The report will return the <N> worst paths based on the specified value, greater than or equal to 1. The default setting is 1.

`-unique_pins` - (Optional) Show only one timing path for each unique set of pins.

`-path_type <arg>` - (Optional) Specify the path data to output in the bus skew report. The default format is full_clock_expanded. The valid path types are:

- short - Displays the startpoints and endpoints with calculated timing values.

- full - Displays the full timing path, including startpoints, through points, and endpoints.

- full_clock - Displays full clock paths in addition to the full timing path.

- full_clock_expanded - Displays full clock paths between a master clock and generated clocks in addition to the full_clock timing path. This is the default setting.

`-sort_by_slack` - (Optional) Sort the Summary and Per Constraint sections of the report by the slack value.

`-input_pins` - (Optional) Show input pins in the timing path report. For use with `-path_type` full, full_clock, and full_clock_expanded.

`-no_header` - (Optional) Do not write a header to the report.

`-significant_digits <arg>` - (Optional) The number of significant digits in the output results. The valid range is 0 to 3. The default setting is 3 significant digits.

`-file <arg>` - (Optional) Write the report into the specified file. The specified file will be overwritten if one already exists, unless `-append` is also specified.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-warn_on_violation` - (Optional) Specify that a Critical Warning will be generated by the Vivado Design Suite when the report contains a violation.

`-rpx <arg>` - (Optional) Specify the file name and path of an Xilinx report file (RPX) to write. This is different from writing the report results to a file using the `-file` argument. The RPX file is an interactive report that contains all the report information and can be reloaded into memory in the Vivado Design Suite using the `open_report` command. You should add a `.rpx` file extension to the specified file name, as the Vivado tool will not automatically assign a file extension.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-cells <arg>` - (Option) Generate the report for the specified hierarchical cells. The details of the report will be based on the specified cells rather than the whole design.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example reports the bus skew for the 32 worst signals of each bus skew constraints in the design, reporting 1 path per bit of the bus with the full timing path, including input pins, with timing values:

```
report_bus_skew -max 32 -nworst 1 -path_type full -input_pins
```

See Also

- open_report
- set_bus_skew

Send Feedback

report_carry_chains

Report carry chains

Syntax

```
report_carry_chains [-file <arg>] [-append] [-return_string] [-cell <args>]
    [-max_chains <arg>] [-quiet] [-verbose]
```

Returns

Report

Usage

Name	Description
[-file]	Filename to output results to. (send output to console if -file is not used)
[-append]	Append to existing file
[-return_string]	return report as string
[-cell]	Report Carry Chains only for given cell
[-max_chains]	Number of chains for which report is to be generated Default: 1
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Report

Description

Report the details of the carry chains used by the current open design. The report includes the average depth of all carry chains, as well as the specific depth of each carry chain reported.

By default, the longest carry chain is reported, but the number of chains reported can be specified.

The command returns the carry chain report.

Arguments

`-file <arg>` - (Optional) Write the report into the specified file. The specified file will be overwritten if one already exists.

Send Feedback

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-append` - (Optional) Append the output of the report to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-cell <arg>` - (Optional) Specify a cell to use when analyzing carry chains. When specified, analysis is limited to only the hierarchical cell specified and to any sub-modules of that cell.

`-max_chains <arg>` - (Optional) Number of chains to report. By default the longest carry chain is reported.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example returns the 10 longest carry chains in the design:

```
report_carry_chains -max_chains 10
```

report_cdc

Report the clock domain crossing (CDC) paths in the current design.

Syntax

```
report_cdc [-from <args>] [-to <args>] [-cells <args>] [-details]
    [-summary] [-all_checks_per_endpoint] [-severity <arg>] [-no_header]
    [-show_waiver] [-no_waiver] [-waived] [-file <arg>] [-append]
    [-return_string] [-name <arg>] [-quiet] [-verbose]
```

Usage

Name	Description
[-from]	From clocks
[-to]	To clocks
[-cells]	run report_cdc on the cells
[-details]	report the detail of the CDC timing paths not safely timed
[-summary]	report a summary by clocks of the CDC
[-all_checks_per_endpoint]	report all checks per endpoint
[-severity]	report only the severity specified (Info, Warning or Critical)
[-no_header]	Do not generate a report header
[-show_waiver]	Show the waived paths
[-no_waiver]	Ignore the waiver
[-waived]	Show only the waived paths
[-file]	Filename to output results to. (send output to console if -file is not used)
[-append]	Append the results to file, don't overwrite the results file
[-return_string]	return report as string
[-name]	Output the results to GUI panel with this name
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Report](#), [Timing](#)

Description

This report shows in detail the clock domain crossing (CDC) paths in the current synthesized or implemented design. The command analyzes paths between asynchronous clocks, or clocks with no common period, as well as synchronous paths ignored by the user due to false path or max delay datapath_only exceptions.

By default the `report_cdc` command reports domain crossing between all clocks in the design. However, you can limit the clocks of interest using the `-from` and `-to` options to specify the clock domains of interest.

The `report_cdc` command only reports on paths where both source and destination clocks are defined. You should run the `check_timing` command prior to `report_cdc` to ensure that there are no unconstrained clocks in the design. I/O paths are only covered by `report_cdc` when input or output delay constraints have been specified on the I/O ports.

The severity of the path report could be Critical, Warning or Info depending on the CDC topology identified. An unknown synchronization topology is Critical and needs to be reviewed. A double register synchronizer with missing ASYNC_REG property is a Warning. Clock Enable, MUX, and MUX Hold CDC structures are categorized as Warnings because you should check to ensure that the structure is safe. Other CDC paths are of severity Info.

The `report_cdc` command returns the following information:

- Severity
- Source Clock
- Destination Clock
- CDC Type
- Exceptions
- Endpoints
- Safe
- Unknown
- No ASYNC_REG property

> **IMPORTANT!** *You cannot use the* `set_msg_config` *command to configure the severity of messages returned by the* `report_cdc` *command. This command does not generate messages through the message manager.*

Arguments

`-from` <args> - (Optional) Report clock domain crossing from the specified clock domain. Clocks can be specified by name or as returned by the `get_clocks` command.

`-to` <args> - (Optional) Report clock domain crossing into the specified clock domain. Clocks can be specified by name or as returned by the `get_clocks` command.

`-cells <arg>` - (Option) Generate the report on the specified hierarchical cells. The details of the report will be based on the specified cells rather than the whole design.

Send Feedback

`-details` - (Optional) Provide a detailed report on the timing paths. The detailed report lists a summary table of the CDC paths, and then lists details of the source/destination clock, and the CDC paths for each clock pair.

`-summary` - (Optional) This is the default report returned for the design. The summary report generates a table with a message severity, the source/destination clock pair, safe CDC, and unknown or unrecognized CDC, and the number of path endpoints.

`-all_checks_per_endpoint` - (Optional) Report all violations for each endpoint of a CDC path, instead of only reporting the most critical violation per endpoint. This option works with the `-severity` option to report all violations per endpoint, of the specified severity.

`-severity [ Critical | Warning | Info ]` - (Optional) Report only the CDC paths with the specified severity level.

`-no_header` - (Optional) Eliminate the report header from the results. This can be especially useful when returning the results as a string with `-return_string`.

`-show_waiver` - (Optional) Used with the `-details` argument, this option adds the waived CDC paths to the report, and adds a "Waived" column to the report to indicate which paths have been waived and which have not.

`-no_waiver` - (Optional) Ignore the waivers defined by the `create_waivers` command and report all CDC paths.

`-waived` - (Optional) Causes only the CDC paths waived by the `create_waiver` command to be reported. This returns the actual violation rather than the definition of the waiver, which can be reported by the `report_waivers` command.

`-file <arg>` - (Optional) Write the report into the specified file.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-name <arg>` - (Optional) The name of the Clock Domain Crossing report view to display in the Vivado IDE when run in GUI mode. If the name has already been used in an open report view, that view will be closed and a new report opened.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example reports the clock domain crossings in the current design, including any waived paths, using a verbose report form, and saving the results to a file:

```
report_cdc -details -show_waiver -file C:/Data/cdc_report.txt
```

The following example reports the clock domain crossings from a clock specified by name, to another specified as a clock object:

```
report_cdc -from clk_pin_p -to [get_clocks clk_rx_clk_core]
```

See Also

- all_clocks
- create_clock
- create_generated_clock
- get_clocks
- report_clock_interaction
- report_clock_networks
- report_clocks
- report_timing
- report_timing_summary
- set_clock_groups
- set_false_path

Send Feedback

report_clock_interaction

Report on clock timing paths and unclocked registers

Syntax

```
report_clock_interaction [-from <args>] [-to <args>] [-delay_type <arg>]
    [-setup] [-hold] [-significant_digits <arg>] [-no_header] [-file <arg>]
    [-append] [-name <arg>] [-return_string] [-cells <args>] [-quiet]
    [-verbose]
```

Usage

Name	Description
[-from]	From clocks
[-to]	To clocks
[-delay_type]	Type of path delay: Values: max, min, min_max Default: max
[-setup]	Consider max delay timing paths (equivalent to -delay_type max)
[-hold]	Consider min delay timing paths (equivalent to -delay_type min)
[-significant_digits]	Number of digits to display: Range: 0 to 3 Default: 2
[-no_header]	do not generate a report header
[-file]	Filename to output results to. (send output to console if -file is not used)
[-append]	Append the results to file, don't overwrite the results file
[-name]	Output the results to GUI panel with this name
[-return_string]	Return report as string
[-cells]	run report_clock_interaction on the specified cell(s)
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Report](), [Timing]()

Description

Reports clock interactions and signals that cross clock domains to identify potential problems such a metastability, or data loss, or incoherency, where some visibility into the paths that cross clock domains is beneficial. This command requires an open synthesized or implemented design.

Note: By default the report is written to the Tcl console or STD output. However, the results can also be written to a file or returned as a string if desired.

Send Feedback

Arguments

`-from <args>` - (Optional) The source clocks of the clock pairs relationship between which the clock interactions should be analyzed. You can specify clock names or clock objects.

`-to <args>` - (Optional) The destination clocks of the clock pairs relationship between which the clock interactions should be analyzed. You can specify clock names or clock objects.

`-delay_type <arg>` - (Optional) Specifies the type of delay to analyze when running the clock interaction report. The valid values are min, max, and min_max. The default setting for `-delay_type` is `max`.

`-setup` - (Optional) Check for setup violations. This is the same as specifying `-delay_type max`.

`-hold` - (Optional) Check for hold violations. This is the same as specifying `-delay_type min`.

Note: `-setup` and `-hold` can be specified together, which is the same as specifying `-delay_type min_max`.

`-significant_digits <arg>` - (Optional) The number of significant digits in the output results. The valid range is 0 to 3. The default setting is 2 significant digits.

`-no_header` - (Optional) Do not write a header to the report.

`-file <arg>` - (Optional) Write the report into the specified file. Unless you also specify `-append`, the specified file will be overwritten if it already exists.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-name <arg>` - (Optional) The name of the Clock Interaction Report view to display in the tool GUI mode. If the name has already been used in an open Report view, that view will be closed and a new report opened.

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-cells <arg>` - (Option) Generate the report for the specified hierarchical cells. The details of the report will be based on the specified cells rather than the whole design.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example sets the model for interconnect delay, selects a device speed grade, and then runs `report_clock_interaction`:

```
set_delay_model -interconnect none
set_speed_grade -3
report_clock_interaction -delay_type min_max \
    -significant_digits 3 -name "results_1"
```

The following example returns the clock interactions, writing the report to the GUI, to the specified file, and returns a string which is assigned to the specified variable:

```
set clk_int [report_clock_interaction -file clk_int.txt -name clk_int1 \
-return_string]
```

See Also

- create_clock
- create_generated_clock
- report_clocks
- set_delay_model
- set_speed_grade

report_clock_networks

Report clock networks

Syntax

```
report_clock_networks [-file <arg>] [-append] [-name <arg>]
    [-return_string] [-endpoints_only] [-levels <arg>] [-expand_buckets]
    [-suppress_endpoints <arg>] [-clocks <args>]
    [-unconstrained_roots <args>] [-quiet] [-verbose]
```

Usage

Name	Description
[-file]	Filename to output results to. (send output to console if -file is not used)
[-append]	Append the results to file, don't overwrite the results file
[-name]	Output the results to GUI panel with this name
[-return_string]	return report as string
[-endpoints_only]	dump clock network endpoints only; Not to be used in conjunction with -levels option
[-levels]	expands clock network upto n levels of instances, Value: n > 0; Not to be used in conjunction with -endpoints_only option Default: 0
[-expand_buckets]	expands bucketed endpoints and displays pins; By default, endpoint pins are bucketed by celltype; This option only works in conjunction with -levels option or -endpoints_only option
[-suppress_endpoints]	suppress paths to clock or nonclock endpoint pins; Values: clock, nonclock
[-clocks]	List of clocks for clock network dump; if not specified, all clock networks are dumped
[-unconstrained_roots]	List of unconstrained root pins/ports for clock network dump; if not specified, all unconstrained clock roots are dumped
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Report](), [Timing]()

Description

Reports the network fanout of each clock net in the open synthesized or implemented design. The graphical form of the report, returned when the `-name` argument is specified, provides a hierarchical tree view of the clock network.

Send Feedback

The default report simply specifies the clock net names and the instance pins that are the startpoint of the clock.

The report is returned to the standard output unless the `-file`, `-return_string`, or `-name` arguments are specified.

Arguments

`-file <arg>` - (Optional) Write the clock network report into the specified file. The specified file will be overwritten if one already exists, unless `-append` is also specified.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-name <arg>` - (Optional) Specifies the name of the results to output to the GUI.

`-return_string` - (Optional) Directs the output to a Tcl string. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-endpoints_only` - (Optional) Report the clock network endpoints; bucketed by cell type, and sorted by clock pins and non-clock pins. This option cannot be specified with the `-levels` option.

`-levels <arg>` - (Optional) Expands the clock network through the specified levels of instances. The default value is 0. The specified value can be greater than 0. This reports the clock path to the path endpoints if enough levels are specified. Path endpoints are bucketed by cell type, and sorted by clock pins and non-clock pins. This option cannot be used with the `-endpoints_only` option.

`-expand_buckets` - (Optional) Expand the clock network report to expand all of the endpoints buckets returned by the `-endpoints_only` or `-levels` options, to report all of the clock endpoints.

`-suppress_endpoints [ clock | nonclock ]` - (Optional) Suppress reported paths from clock root to clock pins or non-clock endpoint pins. This option eliminates either clock or non-clock pins from the clock network report.

`-clocks <args>` - (Optional) Specify the clock objects to include in the clock network report. When not specified, the report includes all clocks.

`-unconstrained_roots` <args> - (Optional) For UltraScale devices, and device architectures with clock roots, specify the list of unconstrained clock root pins or ports to include in the clock network report. If this option is not specified, all unconstrained clock roots are reported.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example reports the clock network names and startpoints to the specified file:

`report_clock_networks` `-file C:/Data/ClkNets.txt`

The following example reports the endpoints of the specified clock:

`report_clock_networks` `-endpoints_only -clocks wbClk`

See Also

- create_clock
- delete_clock_networks_results
- get_clocks

report_clock_utilization

Report information about clock nets in design

Syntax

```
report_clock_utilization [-file <arg>] [-append] [-write_xdc <arg>]
    [-cells <args>] [-clock_roots_only] [-return_string] [-name <arg>]
    [-quiet] [-verbose]
```

Returns

Report

Usage

Name	Description
[-file]	Filename to output results to. (send output to console if -file is not used)
[-append]	Append to existing file
[-write_xdc]	file to output clock constraint. File name must be given.
[-cells]	Cells/bel_instances for which to report clock utilization
[-clock_roots_only]	Report only the Clock Root Assignments
[-return_string]	return report as string
[-name]	Output the results to GUI panel with this name
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Report](), [Timing]()

Description

Returns information related to clock nets in the design and clock resource usage on the target device.

The generated clock utilization report can generate placement constraints for the currently placed clock resources. You can use these constraints to preserve the placement of clock resources for future iterations of the design, by using the `-write_xdc` option.

> **IMPORTANT!** *For Ultrascale devices, if the intent is to recreate the current clock placement then use the BUFGCE LOC properties from the written XDC file. However, if the intent is to use the constraints as a starting point for the clocking architecture, while allowing the Vivado Design Suite some flexibility in placing clock resources, use the equivalent CLOCK_REGION properties instead of the BUFGCE LOC properties.*

By default the report is written to the Tcl console or STD output. However, the results can also be written to a file or returned as a string if desired.

Arguments

`-file <arg>` - (Optional) Write the report into the specified file. The specified file will be overwritten if one already exists.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-write_xdc` <filename> - (Optional) Output XDC location constraints for the various clock resources to the specified filename. If the path is not specified as part of the file name the file will be created in the current working directory, or the directory from which the tool was launched.

`-cells <arg>` - (Option) The details of the report will be based on the specified cells rather than the whole design.

`-clock_roots_only` - (Optional) For UltraScale, and device architectures with clock roots, limit the report output to cover just the clock root assignments for each clock net.

Note: This option is not supported for Xilinx 7 series devices.

`-return_string` - (Optional) Direct the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

`-name <arg>` - (Optional) The name of the Clock Utilization report view to display in the Vivado IDE when run in GUI mode. If the name has already been used in an open Report view, that view will be closed and a new report opened.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example returns information about the clock nets in the design and the clock resources utilized on the target device, and writes it to the specified file:

```
report_clock_utilization -file C:/Data/FPGA_Design/clock_util.txt
```

The following example reports the clock nets and clock resource utilization to the standard output, but writes the XDC location constraints to the specified file:

```
report_clock_utilization -write_xdc clock_util_xdc.txt
```

Note: Because the path is not specified as part of the XDC file name, the file will be created in the current working directory, or the directory from which the tool was launched.

See Also

- create_clock
- create_generated_clock

Send Feedback

report_clocks

Report clocks

Syntax

```
report_clocks [-file <arg>] [-append] [-return_string] [-quiet] [-verbose]
    [<clocks>]
```

Usage

Name	Description
[-file]	Filename to output results to. (send output to console if -file is not used)
[-append]	Append the results to file, don't overwrite the results file
[-return_string]	return report as string
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<clocks>]	List of clocks Default: *

Categories

Report, Timing

Description

Returns a table showing all the clocks in a design, including propagated clocks, generated and auto-generated clocks, virtual clocks, and inverted clocks in the current synthesized or implemented design. More detailed information about each clock net can be obtained with the `report_clock_utilization` command.

Note: By default the report is written to the Tcl console or STD output. However, the results can also be written to a file or returned as a string if desired.

Arguments

`-file <arg>` - (Optional) Write the report into the specified file. The specified file will be overwritten if one already exists, unless `-append` is also specified.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<clocks> - (Optional) The clocks to match against the specified patterns. The default pattern is the wildcard '*' which returns all clocks in the design. More than one pattern can be specified to find multiple clocks based on different search criteria.

Examples

The following example returns the name, period, waveform, and sources of the clocks in the current design:

```
report_clocks -file C:/Data/FPGA_Design/clock_out.txt
```

The following example reports the clocks in the design with "Clock" in the name:

```
report_clocks *Clock*
```

See Also

- create_clock
- create_generated_clock
- report_clock_utilization

report_compile_order

Report the compile order by analyzing files and constructing a hierarchy.

Syntax

```
report_compile_order [-fileset <arg>] [-missing_instances] [-constraints]
    [-sources] [-used_in <arg>] [-file <arg>] [-append]
    [-of_objects <args>] [-quiet] [-verbose]
```

Usage

Name	Description
[-fileset]	FileSet to parse to determine compile order
[-missing_instances]	Report missing instances in the design hierarchy
[-constraints]	Report the constraint compile order
[-sources]	Report the source compile order
[-used_in]	Specify the used in filter.
[-file]	Filename to output results to.
[-append]	Append output to existing file
[-of_objects]	Get 'file' objects of these types: 'file fileset ip reconfig_module'.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Project

Description

Report the compilation order of files in the various active filesets: constraints, design sources, and simulation sources.

This command returns the order of file processing for synthesis, implementation, and simulation. The report can be limited by specifying the fileset of interest with `-fileset`, or using the `-constraints` option or `-sources` option.

The -used_in option lets you report the processing order of files used in Synthesis, Simulation, or one of the implementation steps, according to the value of the USED_IN property.

By default the report is returned to the Tcl console, or standard output, but it can also be written to a file.

Send Feedback

Arguments

`-of_objects` <args> - (Optional) Report the files that are associated with the specified filesets, sub-designs, or reconfiguration modules. The default is to search all filesets. When `-compile_order` and `-used_in` are specified, the `-of_objects` switch will only accept a single fileset, or a single sub-design such as an IP core, Block Design, or DSP design. A sub-design is also known as a composite file.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-fileset` <arg> - (Optional) Limit the report to the specified fileset.

`-missing_instances` - (Optional) Return the list of cells that are missing source files in the current or specified fileset.

`-constraints` - (Optional) Report the compilation order of the constraint files in the current design, including constraints for any IP in the design.

`-sources` - (Optional) Report the compilation order of files found in the `sources_1` fileset of design sources.

`-used_in` <arg> - (Optional) Accepts one of the enumerated values of the `USED_IN` property for files, and returns files matching the specified value. Valid values for this option include the following: synthesis, simulation, or implementation. Refer to the *Vivado Design Suite Properties Reference Guide (UG912)* for information on the USED_IN property and its supported values.

`-file` <arg> - (Optional) Write the report into the specified file. The specified file will be overwritten if one already exists, unless `-append` is also specified.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example reports the compilation order of the active filesets in the current design:

```
report_compile_order
```

The following returns a list of cells with missing source files in the current design, and appends the report to the specified file:

```
report_compile_order -missing_instances -file C:/Data/report1.txt -append
```

The following command lists the compile order of the files in the active constraint set:

```
report_compile_order -constraints
```

See Also

- current_fileset
- get_files
- update_compile_order

report_conditions

Print details of the given condition objects

Syntax

```
report_conditions [-quiet] [-verbose] [<ConditionObjs>...]
```

Returns

Prints name, id, condition_expression and commands of each condition object on the console.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<ConditionObjs>]	ConditionObjs, id's or names

Categories

Simulation

Description

Report a specific simulation condition, or report all conditions in the current simulation. You must have an open simulation for this command to return anything.

Conditions can be defined prior to starting the simulation. When a condition is added, the simulator evaluates the condition expression anytime a signal change is detected. When a specified condition expression becomes TRUE, the condition commands are run.

This command returns the condition identifier, expression, commands, and names of conditions, or returns an error if the command fails.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<ConditionObjs> - (Optional) Specifies one or more condition identifiers in the current simulation to report. The condition identifiers are returned by the `add_condition` command when the condition is defined. BY default the command reports all conditions.

Examples

The following example reports conditions in the current simulation. The condition identifier, expression, commands, and names are reported:

```
report_conditions
#2: condition2
     Expression: {/testbench/reset == 0 }
     Command:    {
puts "Condition Reset was encountered at [current_time]. \
   Stopping simulation."
stop }
     Name:       resetLow
#3: condition3
     Expression: {/testbench/leds_n == X000 }
     Command:    {
puts "Condition LED Unknown was encountered at [current_time]. \
   Stopping simulation."
stop }
     Name:       ledUnknown
```

See Also

- add_condition
- remove_conditions

Send Feedback

report_config_implementation

Report Implementation flow Config param

Syntax

```
report_config_implementation [-file <arg>] [-force] [-append]
    [-return_string] [-quiet] [-verbose]
```

Returns

Report

Usage

Name	Description
[-file]	Filename to output results to. Send output to console if -file is not used.
[-force]	Overwrite existing file
[-append]	Append to existing file
[-return_string]	return report as string
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Report

Description

Report the user-definable implementation flow configuration parameters for the implementation processes. These are parameters that can be modified using the `config_implementation` command.

This command returns the selected report or the name of the output file, or returns an error if it fails.

Arguments

`-file <arg>` - (Optional) Write the configuration parameters report to the specified file. The specified file will not be overwritten if one already exists, unless `-force` is also specified.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-force` - (Optional) Overwrite the specified file if it already exists.

Send Feedback

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

This example reports the user-configurable implementation parameters to the Tcl console:

```
report_config_implementation
```

See Also

- config_implementation

report_config_timing

Report settings affecting timing analysis

Syntax

```
report_config_timing [-file <arg>] [-append] [-name <arg>] [-return_string]
    [-all] [-no_header] [-rpx <arg>] [-quiet] [-verbose]
```

Usage

Name	Description
[-file]	Output the results to file
[-append]	Append the results to file, don't overwrite the results file
[-name]	Output the results to GUI panel with this name
[-return_string]	return report as string
[-all]	report all configuration settings (by default, only the typically important settings are reported
[-no_header]	do not generate a report header
[-rpx]	Filename to output interactive results to.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Report](), [Timing]()

Description

Report the configuration of timing constraints of the current design.

By default the report is abbreviated, containing only a few key timing constraints. Use the `-all` argument to return all timing related configuration.

Arguments

`-file <arg>` - (Optional) Write the timing constraints configuration report into the specified file. The specified file will be overwritten if one already exists, unless `-append` is also specified.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

Send Feedback

`-name <arg>` - (Optional) The name of the results to output to the GUI.

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-all` - (Optional) Reports the state of all timing related attributes and constraints in the design. By default, only a limited set of important timing attributes is reported.

`-no_header` - (Optional) Disables the report header. By default the report includes a header that lists:

- Report Type - timer_configuration.
- Design - The top module of the design.
- Part - The device, package, and speed grade of the target part.
- Version - The version of software used to create the report
- Date - The date of the report.
- Command - The command options used to create the report.

`-rpx <arg>` - (Optional) Specify the file name and path of an Xilinx report file (RPX) to write. This is different from writing the report results to a file using the `-file` argument. The RPX file is an interactive report that contains all the report information and can be reloaded into memory in the Vivado Design Suite using the `open_report` command. You should add a `.rpx` file extension to the specified file name, as the Vivado tool will not automatically assign a file extension.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Send Feedback

Examples

The following example reports the current timing configuration, returns the information as a string, and sets that string into the specified Tcl variable:

```
set timeConfig [report_config_timing -all -no_header -return_string]
puts $timeConfig
```

See Also

- delete_timing_results

report_constant_paths

Report constant paths

Syntax

```
report_constant_paths [-max_paths_per_endpoint <arg>] [-column_style <arg>]
    [-file <arg>] [-append] [-no_header] [-of_objects <args>]
    [-return_string] [-quiet] [-verbose] [<objects>]
```

Usage

Name	Description
[-max_paths_per_endpoint]	number of branches per endpoint to report traces for Default: 1
[-column_style]	style for path report columns: Values: variable_width, anchor_left, fixed_width Default: variable_width
[-file]	Filename to output results to. (send output to console if -file is not used)
[-append]	Append the results to file, don't overwrite the results file
[-no_header]	Do not generate a report header
[-of_objects]	Report constant paths for these constant paths
[-return_string]	return report as string
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<objects>]	endpoint pins or instances

Categories

[Report](), [Timing]()

Description

Reports constant path objects that meet the specified criteria.

Arguments

`-max_paths_per_endpoint` – Specify maximum number of paths to report per constant endpoint object.

Note: By default, 1 path per endpoint has been reported.

`-column_style [ variable_width | anchor_left | fixed_width ]` - (Optional) Specify the format of the timing path portion of the timing report output.

Note: The default format is `anchor_left`.

Send Feedback

`-file <arg>` - (Optional) Write the report into the specified file. The specified file will be overwritten if one already exists, unless `-append` is also specified.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

`-no_header` – (Optional) Will not print header which contains information about tool, system, device, and design on top of the report.

`-of_objects` – (Optional) Report constant paths for specified paths. For object retrieving `get_constant_paths` command can be used.

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns `TCL_OK` regardless of any errors encountered during execution.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

`<objects>` - (Required) Design cell(s) or pin(s) object list to report.

Examples

Report constant paths of the specified pin.

```
report_constant_paths [get_pins cpuEngine/or1200_cpu/or1200_sprs/esr/D]
```

Output a report into the specified file with fixed column style and in append mode.

```
report_constant_paths [get_pins cpuEngine/or1200_cpu/or1200_sprs/esr/D] -
file filename.txt -append -column_style fixed
```

Report constant paths of the specified cell.

```
report_constant_paths [get_cells cpuEngine/or1200_cpu/or1200_sprs/esr]
```

See Also

- get_constant_paths

Send Feedback

report_control_sets

Report the unique control sets in design

Syntax

```
report_control_sets [-file <arg>] [-append] [-hierarchical]
    [-hierarchical_depth <arg>] [-sort_by <args>] [-cells <args>]
    [-return_string] [-quiet] [-verbose]
```

Returns

Report

Usage

Name	Description
[-file]	Filename to output results to. (send output to console if -file is not used)
[-append]	Append to existing file
[-hierarchical]	Generates text-based hierarchical report.
[-hierarchical_depth]	Specifies the depth level for textual hierarchical report Default: 0
[-sort_by]	Sort criterion: can be used only when -verbose is used. Options are clk, clkEn, set. Ex: report_control_sets -verbose -sort_by {clk clkEn}
[-cells]	Cells/bel_instances for which to report control sets
[-return_string]	return report as string
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Report](#)

Description

Report the control sets of the current design.

Control sets are the list of control signals (Clock, CE, SR) for SLICE registers and LUTs. Registers must belong to the same control set in order to be packed into the same device resource. Registers without a control signal cannot be packed into devices with registers having control signals. A high number of control sets can cause difficulty fitting the device and can cause routing congestion and timing issues.

By default the `report_control_sets` command returns an abbreviated report indicating only the number of unique control sets. However, the `-verbose` arguments returns a detailed report of all control sets, for either the whole design or for the specified cells.

Arguments

`-file <arg>` - (Optional) Write the report into the specified file. The specified file will be overwritten if one already exists.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-hierarchical` - (Optional) Generate a hierarchical report of control sets in the design.

`-hierarchical_depth <args>` - (Optional) Specifies the depth of the hierarchy to report when the `-hierarchical` argument is enabled. The default value is 0, which reports the full design hierarchy.

`-sort_by <args>` - (Optional) Sort the detailed report generated by the `-verbose` argument according to the specified criteria. Valid sort criteria are: `clk`, `clkEn`, and `set`.

Note: The `-sort_by` argument is used with `-verbose`.

`-cells <args>` - (Optional) Report control sets for the specified cell objects.

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example reports the control sets of the current design, sorted by the clk and clkEn signals:

```
report_control_sets -verbose -sort_by {clk clkEn}
```

The following example reports the control sets of the specified cells, sorted by clk and set:

```
report_control_sets -verbose -sort_by {clk set} -cells [get_cells usb*]
```

report_datasheet

Report data sheet

Syntax

```
report_datasheet [-significant_digits <arg>] [-file <arg>] [-append]
    [-return_string] [-sort_by <arg>] [-name <arg>] [-show_all_corners]
    [-show_oe_timing] [-group <args>] [-rpx <arg>] [-quiet] [-verbose]
```

Usage

Name	Description
[-significant_digits]	Number of digits to display: Range: 0 to 3 Default: 3
[-file]	Filename to output results to. (send output to console if -file is not used)
[-append]	Append the results to file, don't overwrite the results file
[-return_string]	return report as string
[-sort_by]	Sorting order: Values: clock, port Default: clock
[-name]	Output the results to GUI panel with this name
[-show_all_corners]	provide all corners
[-show_oe_timing]	show output enable (tristate) timing
[-group]	List of output ports for skew calculation
[-rpx]	Filename to output interactive results to.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Report](#), [Timing](#)

Description

Create a "datasheet" report for the current design. Reports setup and hold times of input I/Os in relation to clocks, max/min delays from clocks to output pads, skews of input/ output buses.

The datasheet report has the timing characteristics of a design at the package balls/pads, including the package trace flight times. To disable flight times use the following command:

```
config_timing_analysis -disable_flight_delays true
```

The source synchronous output skew can be automatically calculated by the Vivado Design Suite by using the `-group` switch for `report_datasheet` and grouping together all the ports of the data bus including the sourced clock output port. The sourced clock output port must be first in the group list. For example:

```
report_datasheet -file output_filename -group [get_ports \
    {clock_port data_bit[0] data_bit[1] data_bit[2]}]
```

Arguments

`-significant_digits <arg>` - (Optional) Number of digits to display from 0 to 3. The default is 3.

`-file <arg>` - (Optional) Write the report into the specified file. The specified file will be overwritten if one already exists, unless `-append` is also specified.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-sort_by [ port | clock ]` - (Optional) Sorting order. Valid values are clock or port. The default is to sort the report by clocks.

`-name <arg>` - (Optional) The name of the Datasheet report view to display in the Vivado IDE when run in GUI mode. If the name has already been used in an open Report view, that view will be closed and a new report opened.

`-show_all_corners` - (Optional) Report all process corners.

`-show_oe_timing` - (Optional) Report max and min values for paths using output enable, showing both ON and OFF states.

`-group [get_ports {xxx1 xxx2 ... xxxN}]` - (Optional) Allows you to define your own custom group of ports for analysis. This option requires a list of port objects as returned by the `get_ports` command. The first port in the list will be used as the reference for skew calculation. In most cases, this will be a clock port of a source synchronous output interface. Multiple groups can be specified, each with its own reference clock port. When `-group` is not specified the timer automatically finds the group of output ports based on the launching clock, and reports skew based on that clock.

`-rpx <arg>` - (Optional) Specify the file name and path of an Xilinx report file (RPX) to write. This is different from writing the report results to a file using the `-file` argument. The RPX file is an interactive report that contains all the report information and can be reloaded into memory in the Vivado Design Suite using the `open_report` command. You should add a `.rpx` file extension to the specified file name, as the Vivado tool will not automatically assign a file extension.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example returns the datasheet sorted by ports, for all process corners:

```
report_datasheet -sort_by port -show_all_corners
```

The following example reports the datasheet with the skew calculation for two groups of ports, with the first port of each group providing the reference for the skew calculation for that group. In this example, CLK0OUT is the forwarded clock for DATA0-4 and CLK1OUT is the forwarded clock for DATA4-7:

```
report_datasheet -file ds.txt -group [get_ports \
   {CLK0OUT DATA0 DATA1 DATA2 DATA3}] \
   -group [get_ports {CLK1OUT DATA4 DATA5 DATA6 DATA7}]
```

See Also

* get_ports

Send Feedback

report_debug_core

Report details on debug cores

Syntax

```
report_debug_core [-file <arg>] [-append] [-return_string] [-full_path]
    [-quiet] [-verbose]
```

Usage

Name	Description
[-file]	Filename to output results to. (send output to console if -file is not used)
[-append]	Append the results to file, don't overwrite the results file
[-return_string]	Return report as a string
[-full_path]	Display full hierarchical net path in report
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Report](#), [Debug](#)

Description

Writes a report of the various Vivado device tool debug cores in the current project, and the parameters of those cores. Debug cores can be added to a project using the `create_debug_core` command.

Note: By default the report is written to the Tcl console or STD output. However, the results can also be written to a file or returned as a string if desired.

Arguments

`-file <arg>` - (Optional) Write the report into the specified file. The specified file will be overwritten if one already exists.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

Send Feedback

`-return_string` - (Optional) Return report as a string. This argument can not be used with the `-file` argument.

`-full_path` - (Optional) Display the net names using the full hierarchical path of the net.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example writes the debug core report to the specified file name at the specified location:

```
report_debug_core -file C:/Data/FPGA_Design/project_1_cores.txt
```

See Also

- create_debug_core

report_design_analysis

Report Design Analysis

Syntax

```
report_design_analysis [-file <arg>] [-csv <arg>] [-append]
    [-return_string] [-complexity] [-cells <args>] [-bounding_boxes <args>]
    [-hierarchical_depth <arg>] [-congestion] [-min_congestion_level <arg>]
    [-timing] [-setup] [-hold] [-show_all] [-full_logical_pin]
    [-routed_vs_estimated] [-logic_level_distribution]
    [-logic_level_dist_paths <arg>] [-min_level <arg>] [-max_level <arg>]
    [-return_timing_paths] [-of_timing_paths <args>] [-max_paths <arg>]
    [-extend] [-routes] [-end_point_clocks <args>] [-logic_levels <arg>]
    [-qor_summary] [-json <arg>] [-name <arg>] [-no_pr_attribute] [-quiet]
    [-verbose]
```

Usage

Name	Description
[-file]	Filename to output results to. (send output to console if -file is not used)
[-csv]	Write detailed tables to the CSV filename specified
[-append]	Append the results to file, don't overwrite the results file
[-return_string]	Return report as string
[-complexity]	Finds the interconnection complexity (Rent) of the design
[-cells]	Report analysis information for given list of cells
[-bounding_boxes]	Report interconnection complexity (Rent) for given list of bounding boxes Default: empty
[-hierarchical_depth]	Hierarchical depth option for -complexity Default: 1
[-congestion]	Reports congestion of the design
[-min_congestion_level]	Minimum congestion level for reporting router congestion, integer value between 3 and 8 Default: 5
[-timing]	Reports characteristics of critical path
[-setup]	Reports characteristics of critical SETUP path
[-hold]	Reports characteristics of critical HOLD path
[-show_all]	Adds more characteristics to the timing characteristics report
[-full_logical_pin]	Display hierarchical pin names in the report
[-routed_vs_estimated]	Reports relevant characteristics of critical path in estimated mode
[-logic_level_distribution]	Reports logic level distribution
[-logic_level_dist_paths]	Number of critical paths for analyzing logic level distribution used along with -logic_level_distribution Default: 1000
[-min_level]	Group all paths with logic levels <min_level-1> and below into a single bin, value passed must be at least 1 Default: Not Used

Send Feedback

Name	Description
[-max_level]	Group all paths with logic levels <max_level+1> and above into a single bin, where <max_level> must be the greater of zero or <min_level + 1> if -min_level is used Default: Not Used
[-return_timing_paths]	Returns timing path objects
[-of_timing_paths]	Reports characteristics for these paths
[-max_paths]	Number of paths to consider for -timing option Default: 1
[-extend]	Reports characteristics of worst path before the start point of critical path and worst path after the end of the critical path
[-routes]	Reports distribution with respect to Routes instead of logic levels
[-end_point_clocks]	Returns timing path objects filtered by endpoint clock names as passed to this option
[-logic_levels]	Returns timing path objects bucketed under the bin name as passed to this option
[-qor_summary]	Design Flow summary
[-json]	Write the report in JSON format to the filename specified
[-name]	Output the results to GUI panel with this name
[-no_pr_attribute]	Report without PR attributes
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Report, Timing

Description

Provides timing data on critical path characteristics and complexity of the design to help identify and analyze problem areas that are subject to timing closure issues and routing congestion. For more information on this command refer to the *Vivado Design Suite User Guide: Design Analysis and Closure Techniques* (UG906).

The report_design_analysis command currently has three modes of operation:

- Timing: reports timing and physical characteristics of timing paths.

- Complexity: analyzes the design for routing complexity and LUT distribution.

- Congestion: analyzes the design for routing congestion.

In timing mode, the command calls the static timing engine to analyze critical path data and report the characteristics of each path. The path characteristics include important elements such as clock skew, placement obstacles such as crossing clock regions, and physical constraints such as Pblocks and LOCs. The list of paths can be extended to include a number of top critical paths or specific paths can be analyzed by providing timing path objects to the command. The reports can also be extended to show the paths preceding and following the critical path.

The following are definitions of the characteristics of the paths reported in timing mode:

- PATH_TYPE: either SETUP or HOLD.

- REQUIREMENT: delay requirement from static timing analysis.

- PATH DELAY: data path delay from static timing analysis.

- LOGIC DELAY: the portion of the PATH DELAY attributed to logic on the path.

- NET DELAY: the portion of the PATH DELAY attributed to wires on the path. Note that the net delay is based on the estimated or actual routing delay as specified by the `set_delay_model` command.

- CLOCK SKEW: difference in delay between the source and destination clocks.

- SLACK: path timing slack from static timing analysis.

- CLOCK RELATIONSHIP: SAME_CLOCK or RELATED_CLOCK. Helps identify potentially missed inter-clock constraints.

- TIMING EXCEPTION: the timing exceptions, like `set_false_path` or `set_multicycle_path`, that are assigned to the path.

- LOGIC LEVELS: number of logic levels between the source and destination, reported when the `-logic_level_distribution` is specified.

- LOGICAL PATH: shorthand notation showing the ordered list of cells in the path including the start point and end point.

  Note: For Partial Reconfiguration (PR) designs, the logical path is appended to identify the cell as belonging to a reconfigurable partition (:RP#), or to the static region of the design (:S). A translation table at the bottom of the report maps :RP# to a specific reconfigurable partition.

- START POINT CLOCK: the clock domain of the start point of the path.

- END POINT CLOCK: the clock domain of the end point of the path.

- START POINT PIN PRIMITIVE: the library cell and pin of the start point of the path.

- END POINT PIN PRIMITIVE: the library cell and pin of the end point of the path.

- START POINT PIN: the instance and pin name of the start point.

- END POINT PIN: the instance and pin name of the end point.

- COMB DSP: number of combinational DSP blocks in the path.

- DOA REG: the number of DOA registers on the path.

- DOB REG: the number of DOB registers on the path.

- MREG: the number of MREG registers on the path.

- PREG: the number of PREG registers on the path.

- BRAM CROSSINGS: number of block RAM columns traversed by the path.

- DSP CROSSINGS: number of DSP block columns traversed by the path.

- IO CROSSINGS: number of IO columns traversed by the path.

- CONFIG CROSSINGS: the number of CONFIG tile traversed by the path.

- SLR CROSSINGS: number of SLRs traversed by the path.

- BOUNDING BOX SIZE: the rectangular area covered by the critical path, measured in RPM GRID units which are based on the device RPM_X (horizontal) and RPM_Y (vertical) site coordinates. Because different sites (slices, DSP, block RAM, etc.) have different sizes, each site has unique RPM_X and RPM_Y properties to pinpoint its location within the device.

- CLOCK REGION DISTANCE: An ordered pair showing the number of clock regions traversed in the horizontal and vertical directions from path startpoint to endpoint. Minimizing clock region crossings can improve critical path delay and clock skew.

  ○ Example 1: A critical path begins in clock region X1Y1 and ends in clock region X3Y3, resulting in a CLOCK_REGION_DISTANCE of (2, 2).

  ○ Example 2: a critical path begins in clock region X2Y1 and ends in X0Y0, resulting in a CLOCK_REGION_DISTANCE of (-2, -1).

- PBLOCKS: number of Pblocks traversed by the path.

- HIGH FANOUT: the greatest fanout of a net in the path.

- CUMULATIVE FANOUT: the total fanout on the path.

- DONT TOUCH: number of cells in the path with DONT_TOUCH value of TRUE. A value of TRUE for DONT_TOUCH on a cell prevents it from being optimized, disabling potentially beneficial optimizations such as phys_opt_design replication.

- MARK DEBUG: number of cells in the path with a MARK_DEBUG value of TRUE. By default a net with MARK_DEBUG has DONT_TOUCH set to TRUE which disables optimization on that net. The DONT_TOUCH can be set to FALSE to enable optimization and potentially improve timing.

- FIXED LOC: number of placed cells in the path with an IS_LOC_FIXED value of TRUE. FIXED cells cannot be moved by either place_design or phys_opt_design.

- FIXED ROUTE: number of routed nets in the path with IS_ROUTE_FIXED value of TRUE. FIXED routes cannot be ripped up and rerouted by route_design.

- HOLD FIX DETOUR: the amount of routing detour provided to fix hold timing to post-route critical paths.

- COMBINED LUT PAIRS: number of LUT cells in the path that have been combined with other LUT cells into the same LUT BEL to use both the O6 and O5 outputs. LUT cells that have been combined with LUTNM, HLUTNM, or SOFT_HLUTNM can be uncombined and re-placed by setting their HLUTNM properties to an empty string. This allows exploring LUT combining and un-combining effects on timing and congestion reduction.

- The following fields are reported for Partial Reconfiguration (PR) designs. Refer to the *Vivado Design Suite User Guide: Dynamic Function eXchange* (UG909) for more information.

- ◦ PR PATH TYPE: Specifies the path as being completely in the static region, completely in a reconfigurable partition (RP), or as crossing the boundary between regions. The delay elements for the timing path are also broken down between the regions .

- ◦ STATIC CROSSINGS: Reports the number of times a reconfigurable partition (RP) path crosses into the static region.

- ◦ RP CROSSINGS: Reports the number of times a static region path crosses into a reconfigurable partition (RP) region.

- ◦ BOUNDARY FANOUT: Reports the fanout of a boundary path at the PPLOC to its downstream loads .

In complexity mode, the command performs complexity analysis of the current design and reports the Rent Exponent which is a measure of complexity, the Average Fanout, and a Primitive Histogram. The analysis can be performed on the top-level design or recursively on hierarchical levels of the design, with the ability to control the level of recursion.

The following are definitions of the characteristics reported in complexity mode:

- Rent: The Rent exponent, as defined by Rent's rule, is a measure of interconnect complexity in a netlist. Higher Rent indicates higher complexity and greater difficulty to avoid routing congestion. Most designs have a Rent in the 0.5 to 0.6 range. A Rent value of 0.65 is considered high and 0.85 is considered very high.

- Average Fanout: This is the average fanout of a logic cell in the design, excluding global buffers. Higher average fanout may result in more difficulty for placement and routing. While absolute values may not predict difficultly, relative values between designs or between hierarchical levels may be more indicative.

- Primitive Histogram: This displays the totals of certain primitive types used in the design. A high Rent may be caused by a predominance of LUT6 cells. If there are many more LUT6 than other size LUTs, the Rent may be reduced by adopting a more area-focused synthesis strategy.

TIP: *The complexity characteristics may not always predict routing congestion but can be used to pinpoint problem areas when congestion issues occur.*

In congestion mode the command analyzes the design and provides metrics to help you alleviate routing congestion. Using the results from the `report_design_analysis` command, you can change placement to avoid specific routing hot spots.

The command returns the file created, or returns the analysis results to the Tcl console, or returns an error if it fails.

Arguments

`-file <arg>` - (Optional) Write the analysis results into the specified file. The specified file will be overwritten if one already exists unless `-append` is also specified.

`-csv <arg>` - (Optional) Write a CSV file that shows the timing paths and analysis features. The specified file will be overwritten if one already exists.

Note: If the path is not specified as part of the file name for either `-file` or `-csv`, the file will be written into the current working directory that can be identified using the `[pwd]` Tcl command.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-complexity` - (Optional) Perform complexity analysis of the design and report the Rent Exponent, Average Fanout, and Primitive Histogram.

> 💡 **TIP:** *The `-complexity` option can be specified with `-cells` and `-hierarchical_depth` to control the analysis of the design or cells.*

`-cells <arg>` - (Optional) Specify the cells to use when reporting design analysis. Cells should be hierarchical modules of the design. Use this option to specify a cell other than the top cell for analysis. Cells can be specified by name, or as objects returned by the `get_cells` command. Cannot be used with `-hierarchical_depth`.

`-bounding_boxes <args>` - (Optional) Specify regions of the device to analyze for complexity. This limits the area and reduces the time required for analysis. The argument is specified as a pair of device tiles representing the "lower-left:upper-right" corners of the bounding box. Multiple windows can be specified as follows:

```
-bounding_boxes {  "CLE_M_X21Y239:CLEL_R_X28Y254" \
    "CLEL_R_X18Y171:CLE_M_X26Y186" }
```

> ⭐ **IMPORTANT!** *A space is required between the '{' and the start of the bounding box as shown above.*

`-hierarchical_depth <arg>` - (Optional) Specify that the complexity analysis should be performed on the hierarchy of the design, rather than just the top-level of the design, or the specified cell. In this case, the hierarchy of the design or cell will be examined to the specified depth. The default setting is 1. Requires the `-complexity` option.

`-congestion` - (Optional) Reports the vertical and horizontal routing congestion of the placed design on a per tile basis. This option can only be run after implementation, or after `place_design`. Congestion analysis is performed on the whole design, even if `-cells` is specified for complexity analysis.

`-min_congestion_level <arg>` - (Optional) Specifies the minimum congestion level to show congestion in the design. The minimum supported value is 3 as the report does not have any information on congestion windows under level 3. The default value is 5 if the option is not specified.

`-timing` - (Optional) Runs design analysis in timing mode, to report setup and hold characteristics of critical paths. This is the default analysis run by the `report_design_analysis` command when no other options are specified.

> 💡 **TIP:** *The `-timing` option can be specified with `-setup`, `-hold`, `-of_timing_paths`, `-max_paths`, and `-extend`, but can not be used with `-cells` or `-hierarchical_depth`.*

`-setup` - (Optional) Limit the reporting scope to only setup path characteristics. Requires the `-timing` option.

`-hold` - (Optional) Limit the reporting scope to only hold path characteristics. Requires the `-timing` option.

> ⭐ **IMPORTANT!** *When `-hold` is used with the `-extend` option this generates two side-by-side reports: the setup report showing the worst setup paths on the left, and the corresponding hold report for those paths on the right. This shows hold characteristics side-by-side with the setup characteristics of the same start and endpoints to make it easier to see if hold and setup fixing are affecting one another.*

`-show_all` - (Optional) Adds more characteristics to the timing characteristics report.

`-full_logical_pin` - (Optional) Output the full hierarchical pin name in the design analysis report.

`-routed_vs_estimated` - (Optional) Reports the estimated vs. actual routed delays side-by-side for the same path.

> ⭐ **IMPORTANT!** *This requires the `set_delay_model -interconnect` to be set to "actual" to provide both the estimated and actual delays.*

`-logic_level_distribution` - (Optional) Reports logic-level distribution of the `-timing` report to help identify the longer paths in the design. The logic levels distribution can be very runtime intensive so it is not reported by default.

> 💡 **TIP:** *Remember that `-logic_level_distribution` is performed on critical timing paths identified by the `-timing` option. If the purpose of analysis is to identify all of the paths with the greatest number of logic levels in the design, you can use `-of_timing_paths` with the `get_timing_paths` command to define the timing paths of interest. In addition, you can use the LOGIC_LEVELS property on timing paths to obtain this information.*

`-logic_level_dist_paths <arg>` - (Optional) This option can be specified with `-logic_level_distribution` to specify the number of critical paths for analyzing logic level distribution. Specified as a value greater than or equal to 1. The default is 1000 paths.

`-min_level <arg>` - (Optional) Group all timing paths with logic levels, or routes, less than the value specified into a single bin. When specified, the <arg> value passed must be at least 1.

`-max_level <arg>` - (Optional) Group all timing paths with logic levels, or routes, greater than the value specified into a single bin. When specified, the <arg> value must be greater than the value of `-min_level`, or can be zero.

`-return_timing_paths` - (Optional) This must be used with the `-end_point_clock` and `-logic_levels` options. Directs the `report_design_analysis` command to return timing_path objects as well as the analysis report. The timing_path objects can be passed to other analysis commands that take timing paths such as `report_timing`, or even passed to the `-of_timing_paths` option of a second `report_design_analysis` command.

> 💡 **TIP:** *If* `-return_timing_paths` *is specified,* `-timing` *will be disabled so that no timing results are returned. If no timing paths match the specified options, the command will return the message "No timing paths returned."*

`-of_timing_paths <args>` - (Optional) Specify a list of timing path objects for analysis. Requires the `-timing` option. The command will report characteristics of the specified paths instead of the most critical paths. You can use the `get_timing_paths` command to provide the timing path objects for `-of_timing_paths`.

`-max_paths <arg>` - Specify the number of paths to report for critical path analysis. The default number of critical paths analyzed is 1. Note, the number of paths returned is affected by other options such as `-end_point_clock`, which can reduce the number of paths returned due to the specified clock.

`-extend` - (Optional) Perform an extended setup analysis of the critical paths. The hold analysis is not performed. Requires the `-timing` option. Three paths are reported for an extended analysis:

- The path ending at critical path start point.

- The critical path.

- The path starting at critical path endpoint.

`-routes` - (Optional) Reports the path distribution with respect to the number of routes per end point clock instead of logic levels. When it is used with the `-of_timing_objects` option, the `get_timing_paths` command must be run with the `-routable_nets` option.

Send Feedback

`-end_point_clocks <arg>` - (Optional) Takes a list of clocks that can be specified by name or by using get_clocks. Limits the timing paths under consideration to only the ones in the specified clock group.

`-logic_levels <arg>` - (Optional) Filters timing paths by the specified number of logic levels. You can specify one logic level and it can be used with `-end_point_clocks` to further limit timing paths based on the clock domain.

`-qor_summary` - (Optional) Include a summary of the QoR for the design.

`-json <arg>` - (Optional) Write out the QoR summary in JSON format.

Note: The `-json` option can only be used with the `-qor_summary` switch.

`-name <arg>` - (Optional) The name of the Design Analysis Report view to display in the tool GUI mode. If the name has already been used in an open Report view, that view will be closed and a new report opened.

`-no_pr_attribute` - (Optional) Report the design analysis without partial reconfiguration (PR) attributes. For Partial Reconfiguration (PR) designs, the logical path is defined as belonging to a reconfigurable partition (:RP#), or to the static region of the design (:S), as well as whether the path is completely in the PR region or S region or crosses between the regions.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

Perform a complexity analysis of the two specified cells.

```
report_design_analysis -complexity -cells {cpuEngine fftEngine}
```

Perform complexity analysis of the specified bounding boxes.

```
report_design_analysis -complexity \
-bounding_boxes { "CLE_M_X21Y239:CLEL_R_X28Y254"
"CLEL_R_X18Y171:CLE_M_X26Y186" }
```

Send Feedback

Get an extended analysis of the worst critical path from the Block RAMs in the design.

```
report_design_analysis -timing -of_timing_paths \
[get_timing_paths -from [all_rams]]
```

Perform complexity analysis for the specified cell, to a depth of two hierarchical levels, and performs timing and congestion analysis on the design.

```
report_design_analysis -complexity -hierarchical_depth 2 -timing -setup \
-hold -max_paths 10 -logic_level_distribution -logic_level_dist_paths 20 -
congestion
```

Use the `report_design_analysis` command to return the timing paths with the specified end point clock and logic levels, and pass those paths to the `report_timing` command for analysis.

```
report_timing -of_objects [report_design_analysis -end_point_clock cpuClk \
-logic_levels 10 -timing -return_timing_paths]
```

See Also

- config_design_analysis
- get_cells
- get_timing_paths
- place_design
- report_timing
- report_timing_summary
- set_delay_model
- set_false_path
- set_multicycle_path

Send Feedback

report_disable_timing

Report disabled timing arcs

Syntax

```
report_disable_timing [-user_disabled] [-column_style <arg>] [-file <arg>]
    [-append] [-cells <args>] [-return_string] [-quiet] [-verbose]
```

Usage

Name	Description
[-user_disabled]	report only user disabled arcs
[-column_style]	style for path report columns: Values: variable_width, anchor_left Default: variable_width
[-file]	Filename to output results to. (send output to console if -file is not used)
[-append]	Append the results to file, don't overwrite the results file
[-cells]	run report_disable_timing on the specified cell(s)
[-return_string]	return report as string
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Report, Timing

Description

Displays a report of timing paths that will be excluded from timing analysis in the current synthesized or implemented design.

The format of the report is organized into columns for "Cell or Port" to define the object associated with the timing path, "From" and "To" to define the timing path, the condition, and the reason for excluding the path from timing. The various reasons for exclusion are as follows:

- constraint - `set_disable_timing` constraint is specified

- constant - Logic constant

- loop - Breaks a logic loop

- bidirect instance path - Feedback path through bidirectional instances

- bidirect net path - Feedback path on nets with bidirectional pins

Note: By default the report is written to the Tcl console or STD output. However, the results can also be written to a file or returned as a string if desired.

Send Feedback

Arguments

`-user_disabled` - (Optional) Report only the timing arcs that have been disabled by the user with the `set_disable_timing` command.

`-column_style [ variable_width | anchor_left ]` - (Optional) Specifies the alignment of the columns in the report. The default is `variable_width` when the option is not specified.

`-file <arg>` - (Optional) Write the report into the specified file. The specified file will be overwritten if one already exists, unless `-append` is also specified.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-cells <arg>` - (Option) The details of the report will be based on the specified cells rather than the whole design.

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example reports all timing paths that will not be included in timing analysis:

```
report_disable_timing
```

Send Feedback

The following example outputs the disable timing report as a string, stores it in a variable, and then puts it to the display:

```
set bad_time [report_disable_timing -return_string]
puts $bad_time
```

See Also

- report_timing
- set_disable_timing

report_drc

Run DRC

Syntax

```
report_drc [-name <arg>] [-upgrade_cw] [-checks <args>] [-ruledecks <args>]
    [-file <arg>] [-rpx <arg>] [-append] [-waived] [-no_waivers]
    [-return_string] [-quiet] [-verbose]
```

Usage

Name	Description
[-name]	Output the results to GUI panel with this name
[-upgrade_cw]	Specifies if report_drc should upgrade all CRITICAL_WARNING violations to ERROR.
[-checks]	DRC checks (see get_drc_checks for available checks)
[-ruledecks]	Containers of DRC rule checks Default: default
[-file]	Filename to output results to. (send output to console if -file is not used)
[-rpx]	Report filename for persisted results.
[-append]	Append the results to file, do not overwrite the results file
[-waived]	Output result is Waived checks
[-no_waivers]	Disable waivers for checks
[-return_string]	return report as string
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[DRC](#), [Report](#), [Timing](#)

Description

Check the current design against a specified set of design rule checks, or rule decks, and report any errors or violations that are found.

The `report_drc` command requires an open design to check the design rules against. The command returns a report with the results of violations found by the design rule checks. Violations are returned as Vivado objects that can be listed with the `get_drc_violations` command, and are associated with cells, pins, ports, nets, and sites in the current design. You can get the cells, nets, and other design objects that are associated with DRC violation objects, using the `-of_objects` option of the `get_cells` command for instance.

Send Feedback

> 💡 **TIP:** *The* `report_drc` *can be multi-threaded to speed the process. Refer to the* `set_param` *command for more information on setting the* `general.maxThreads` *parameter.*

The Vivado tools include a large number of predefined design rule checks to be used by the `report_drc` command. Use the `get_drc_checks` command to list the currently defined design rule checks. You can also create new custom design rule checks using the `create_drc_check` command.

A rule deck is a collection of design rule checks grouped for convenience, to be run at different stages of the FPGA design flow, such as during I/O planning or placement. The tool comes with a set of factory defined rule decks, but you can also create new user-defined rule decks with the `create_drc_ruledeck` command. Use the `get_drc_ruledecks` command to return a list of the currently defined rule decks available for use in the report_drc command.

The `report_drc` command runs a default rule deck when the `-checks` or `-ruledeck` options are not specified. Creating a user-defined DRC automatically adds the new design rule check to the default rule deck.

DRC rules can be enabled or disabled using the `IS_ENABLED` property on the rule check object. If a rule `IS_ENABLED` false, the rule will not be run by the `report_drc` command, whether it is specified directly using `-checks`, or indirectly with `-ruledeck`.

> 💡 **TIP:** *You can reset the properties of a DRC rule to the factory default settings using the* `reset_drc_check` *command.*

You can reset the current results of the `report_drc` command, clearing any found violations, using the `reset_drc` command.

Arguments

`-name <arg>` - (Optional) The name to assign to the results when run in GUI mode.

`-upgrade_cw <arg>` - (Optional) Report all found Critical Warnings as Errors for this report.

`-checks <args>` - (Optional) A list of rule checks to run the DRC report against. All specified rules will be checked against the current design. Rules are listed by their group name or full key. Using the `-checks` option creates a temporary user-defined rule deck, with the specified design rule checks, and uses the temporary rule deck for the run.

Note: `-ruledeck` and `-checks` cannot be used together.

`-ruledecks <arg>` - (Optional) The name of one or more DRC rule decks. A rule deck is a list of DRC rule check names. You can provide the name of a factory DRC rule deck or a user-defined rule deck. The `report_drc` command checks the design against the rules that are added to the given rule deck. Custom rule decks can be defined using the `create_drc_ruledeck` command. Use the `get_drc_ruledecks` command to list the currently defined rule decks.

`-file <arg>` - (Optional) Write the DRC report into the specified file. The specified file will be overwritten if one already exists, unless `-append` is also specified.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-rpx <arg>` - (Optional) Specify the file name and path of an Xilinx report file (RPX) to write. This is different from writing the report results to a file using the `-file` argument. The RPX file is an interactive report that contains all the report information and can be reloaded into memory in the Vivado Design Suite using the `open_report` command. You should add a `.rpx` file extension to the specified file name, as the Vivado tool will not automatically assign a file extension.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-waived` - (Optional) Causes only the DRC checks waived by the `create_waiver` command to be run and reported. This returns the actual violation rather than the definition of the waiver, which can be reported by the `report_waivers` command.

`-no_waivers` - (Optional) Ignore the waivers defined by the `create_waivers` command and report all DRC violations.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example lists the available rule decks. The results include all factory rule decks and all user-defined rule decks.

```
get_drc_ruledecks
```

The following example returns the list of DRC rules defined in the specified rule deck:

```
get_drc_checks -of_objects [get_drc_ruledecks placer_checks]
```

The following examples run the specified DRC rule deck and rules against the current design, and writes the results to the specified file:

```
report_drc -ruledecks placer_checks -file C:/Data/DRC_Rpt1.txt
report_drc -checks {IOCNT-1 IOPCPR-1 IOPCMGT-1 IOCTMGT-1 IODIR-1}  \
    -file C:/Data/DRC_Rpt1.txt -append
```

Note: The `-append` option adds the result of the second `report_drc` command to the specified file.

See Also

- create_drc_check
- create_drc_ruledeck
- create_drc_violation
- create_waiver
- get_cells
- get_drc_checks
- get_drc_ruledecks
- get_drc_violations
- get_nets
- get_pins
- get_ports
- get_sites
- open_report
- reset_drc
- reset_drc_check
- set_param

Send Feedback

report_drivers

Print drivers along with current driving values for an HDL wire or signal object

Syntax

```
report_drivers [-quiet] [-verbose] <hdl_object>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hdl_object>	Which hdl_object to report

Categories

Simulation

Description

The `report_drivers` command prints the name and value of the driving signal, as well as the current value of a signal type HDL object.

Use this command to determine what signal or process is driving the value on a specific HDL signal, or net object. A driver of a signal is the statement in the HDL source file that is performing assignment to the signal.

The output format of `report_drivers` is as follows:

```
Drivers for <hdl_object>
  <Value of HDL Object>: Net <Hierarchical name of the probed signal>
    [ Declared Net : <The declared signal to which the probed signal is
connected>]
    <Value of Driver> : Driver <Hierarchical name of the HDL process
containing
the driver> at <file_name>:<line number>
```

Note: The Declared Net is returned when the probed signal name is different from the hierarchical name of the actual declared signal due to the current scope of the simulation. Each bit of the declared net is printed for the probed signal.

The values of signals returned by the `report_drivers` command depend on the state of the simulation. In the following example, the report is run before and after simulation:

```
current_scope /testbench/dut
report_drivers leds_n[3:0]
Drivers for /testbench/dut/LEDS_n[3:0]
  0 : Net /testbench/dut/LEDS_n[0]
    Declared Net : /testbench/leds_n[3]
    0 : Driver /testbench/dut/line__187 at C:/Data/sources/
sinegen_demo.vhd:187
    0 : Driver /testbench/dut/line__186 at C:/Data/sources/
sinegen_demo.vhd:186
    0 : Driver /testbench/dut/line__185 at C:/Data/sources/
sinegen_demo.vhd:185
    0 : Driver /testbench/dut/line__184 at C:/Data/sources/
sinegen_demo.vhd:184
  0 : Net /testbench/dut/LEDS_n[1]
    Declared Net : /testbench/leds_n[2]
    0 : Driver /testbench/dut/line__187 at C:/Data/sources/
sinegen_demo.vhd:187
    0 : Driver /testbench/dut/line__186 at C:/Data/sources/
sinegen_demo.vhd:186
    0 : Driver /testbench/dut/line__185 at C:/Data/sources/
sinegen_demo.vhd:185
    0 : Driver /testbench/dut/line__184 at C:/Data/sources/
sinegen_demo.vhd:184
  0 : Net /testbench/dut/LEDS_n[2]
    Declared Net : /testbench/leds_n[1]
    0 : Driver /testbench/dut/line__187 at C:/Data/sources/
sinegen_demo.vhd:187
    1 : Driver /testbench/dut/line__186 at C:/Data/sources/
sinegen_demo.vhd:186
    1 : Driver /testbench/dut/line__185 at C:/Data/sources/
sinegen_demo.vhd:185
    1 : Driver /testbench/dut/line__184 at C:/Data/sources/
sinegen_demo.vhd:184
  X : Net /testbench/dut/LEDS_n[3]
    Declared Net : /testbench/leds_n[0]
    0 : Driver /testbench/dut/line__187 at C:/Data/sources/
sinegen_demo.vhd:187
    0 : Driver /testbench/dut/line__186 at C:/Data/sources/
sinegen_demo.vhd:186
    0 : Driver /testbench/dut/line__185 at C:/Data/sources/
sinegen_demo.vhd:185
    0 : Driver /testbench/dut/line__184 at C:/Data/sources/
sinegen_demo.vhd:184
run all
report_drivers leds_n[3:0]
Drivers for /testbench/dut/LEDS_n[3:0]
  0 : Net /testbench/dut/LEDS_n[0]
    Declared Net : /testbench/leds_n[3]
    0 : Driver /testbench/dut/line__187 at C:/Data/sources/
sinegen_demo.vhd:187
    0 : Driver /testbench/dut/line__186 at C:/Data/sources/
sinegen_demo.vhd:186
    0 : Driver /testbench/dut/line__185 at C:/Data/sources/
sinegen_demo.vhd:185
    0 : Driver /testbench/dut/line__184 at C:/Data/sources/
sinegen_demo.vhd:184
  1 : Net /testbench/dut/LEDS_n[1]
    Declared Net : /testbench/leds_n[2]
    0 : Driver /testbench/dut/line__187 at C:/Data/sources/
```

```
sinegen_demo.vhd:187
    0 : Driver /testbench/dut/line__186 at C:/Data/sources/
sinegen_demo.vhd:186
    0 : Driver /testbench/dut/line__185 at C:/Data/sources/
sinegen_demo.vhd:185
    0 : Driver /testbench/dut/line__184 at C:/Data/sources/
sinegen_demo.vhd:184
  0 : Net /testbench/dut/LEDS_n[2]
    Declared Net : /testbench/leds_n[1]
    1 : Driver /testbench/dut/line__187 at C:/Data/sources/
sinegen_demo.vhd:187
    1 : Driver /testbench/dut/line__186 at C:/Data/sources/
sinegen_demo.vhd:186
    1 : Driver /testbench/dut/line__185 at C:/Data/sources/
sinegen_demo.vhd:185
    1 : Driver /testbench/dut/line__184 at C:/Data/sources/
sinegen_demo.vhd:184
  0 : Net /testbench/dut/LEDS_n[3]
    Declared Net : /testbench/leds_n[0]
    0 : Driver /testbench/dut/line__187 at C:/Data/sources/
sinegen_demo.vhd:187
    0 : Driver /testbench/dut/line__186 at C:/Data/sources/
sinegen_demo.vhd:186
    1 : Driver /testbench/dut/line__185 at C:/Data/sources/
sinegen_demo.vhd:185
    0 : Driver /testbench/dut/line__184 at C:/Data/sources/
sinegen_demo.vhd:184
```

Note: Notice the declared net is reported, because the current scope of the simulation is set to a different level than the top-level of the test bench.

This command returns a report of the drivers on the specified objects, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hdl_objects> - Report the drivers of the specified VHDL signals or Verilog wires. The HDL objects can be specified by name, or returned by the `get_objects` command.

Examples

The following example reports the drivers for the HDL objects returned by the `get_objects` command:

```
report_drivers [get_objects leds_n]
```

See Also

- current_scope
- get_objects

Send Feedback

report_environment

Report system information.

Syntax

```
report_environment [-file <arg>] [-format <arg>] [-append] [-return_string]
    [-quiet] [-verbose]
```

Usage

Name	Description
[-file]	Write system information to specified file.
[-format]	Specifies how to format the report. Default is 'text', another option is 'xml'. Only applies if -file is used. If xml output is used, -append is not allowed. Default: text
[-append]	Append report to existing file
[-return_string]	Return report content as a string value
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Report](#)

Description

Report the details of the system environment that the tool is running under. The details of the environment report include: operating system version, CPU, memory, available disk space, and specific settings of various environment variables.

The default is to write the report to the standard output. However, the report can be written to a file instead.

Arguments

`-file <arg>` - (Optional) Write the report into the specified file. The specified file will be overwritten if one already exists, unless `-append` is also specified.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-format [ text | xml ]` - (Optional) The default format of the output report is text. You can also output an XML report. XML output is only valid when `-file` is specified, and cannot be used with `-append`.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example reports the current environment to the specified file:

```
report_environment -file C:/Data/toolEnv.txt
```

report_exceptions

Report timing exceptions

Syntax

```
report_exceptions [-from <args>] [-rise_from <args>] [-fall_from <args>]
    [-to <args>] [-rise_to <args>] [-fall_to <args>] [-through <args>]
    [-rise_through <args>] [-fall_through <args>] [-ignored]
    [-scope_override] [-summary] [-coverage] [-ignored_objects]
    [-count_objects] [-write_merged_exceptions] [-write_valid_exceptions]
    [-no_header] [-file <arg>] [-append] [-return_string] [-name <arg>]
    [-quiet] [-verbose]
```

Usage

Name	Description
[-from]	From pins, ports, cells or clocks
[-rise_from]	Rising from pins, ports, cells or clocks
[-fall_from]	Falling from pins, ports, cells or clocks
[-to]	To pins, ports, cells or clocks
[-rise_to]	Rising to pins, ports, cells or clocks
[-fall_to]	Falling to pins, ports, cells or clocks
[-through]	Through pins, ports, cells or nets
[-rise_through]	Rising through pins, ports, cells or nets
[-fall_through]	Falling through pins, ports, cells or nets
[-ignored]	Only report exceptions which are fully ignored
[-scope_override]	Only report scoped exceptions which are fully or partially overriden
[-summary]	Report a summary of all exceptions
[-coverage]	Report the coverage of all timing exceptions
[-ignored_objects]	Report the list of ignored startpoints and endpoints
[-count_objects]	Report the number of objects in the timing exceptions
[-write_merged_exceptions]	Write merged timing exceptions
[-write_valid_exceptions]	Write timing exceptions with the valid objects only
[-no_header]	Do not generate a report header
[-file]	Filename to output results to. (send output to console if -file is not used)
[-append]	Append the results to file, don't overwrite the results file
[-return_string]	Return report as string
[-name]	Output the results to GUI panel with this name
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Description

Report all timing exceptions applied to setup and hold checks defined by timing constraints in the current design, or report the exceptions on the specified timing paths.

Timing exceptions can be defined by timing constraints such as `set_false_path` or `set_multicycle_path` that change the default assumptions for timing paths in the design.

The exceptions are reported to the standard output by default, but can be redirected to a file or to a Tcl string variable.

Arguments

`-from` <args> - (Optional) A list of start points on the timing path to report exceptions on.

`-rise_from` <args> - (Optional) A list of the start points on the timing path to report exceptions on the rising-edge of the path.

`-fall_from` <args> - (Optional) A list of the start points on the timing path to report exceptions on the falling-edge of the path.

> ⭐ **IMPORTANT!** *Using the `report_exceptions` command with `-from/-through/-to` options only matches timing exceptions that have been defined with the same `-from/-through/-to` command line options. The specified patterns can be different but the cell, pin, and port objects must also be the same to be reported as an exception.*

`-to` <args> - (Optional) A list of the end points for the timing path to report exceptions on.

`-rise_to` <args> - (Optional) A list of the end points on the timing path to report exceptions on the rising-edge of the path.

`-fall_to` <args> - (Optional) A list of the end points on the timing path to report exceptions on the falling-edge of the path.

`-through` <args> - (Optional) A list of pins, cell, or nets through which the timing path passes.

`-rise_through` <args> - (Optional) A list of pins, cell, or nets through which the rising-edge timing path passes.

`-fall_through` <args> - (Optional) Specifies the list of pins, cell, or nets through which the falling-edge timing path passes.

Send Feedback

`-ignored` - (Optional) Report timing path exceptions in the current design that are fully ignored by the Vivado timing engine. Ignored constraints could be the result of an incorrectly defined constraint, or of missing design objects, or of non-existent path. Timing exceptions that are partially overridden are not reported with this option.

`-scope_override` - (Optional) Report scoped timing path exceptions in the current design that are fully or partially overridden by top-level constraints. This option does not report when the scoped constraints are overridden by other scoped constraints whether from the same scope or from a different scope.

`-summary` - (Optional) Report a summary of all timing exceptions.

`-coverage` - (Optional) Report the coverage of all timing exceptions. Coverage is expressed as a percentage showing the number of pins reached by the timing exception compared to the number of pins specified by the `-from/-through/-to` options.

`-ignored_objects` - (Optional) Report the start points and endpoint objects that are part of timing path exceptions, and ignored by the timing engine.

`-count_objects` - (Optional) Returns the number of objects in the exceptions report.

`-write_valid_exceptions` - (Optional) Write only the valid timing exceptions to the report. Valid exceptions have valid start points and endpoints.

`-write_merged_exceptions` - (Optional) Write the merged timing exceptions. Merged timing exceptions include both the valid and invalid timing exceptions.

`-no_header` - (Optional) Do not write a header to the report.

`-file <arg>` - (Optional) Write the report into the specified file. By default the timing exceptions are reported to the standard output, or the Tcl console. The specified file will be overwritten if one already exists, unless `-append` is also specified.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-name <arg>` - (Optional) The name of the report view to display in the Vivado IDE when run in GUI mode. If the name has already been used in an open report view, that view will be closed and a new report opened.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

Report all timing exceptions in the current design.

```
report_exceptions
```

Report all timing exceptions ignored or overridden in the current design.

```
report_exceptions -ignored
```

See Also

- create_clock
- create_generated_clock
- report_timing
- report_timing_summary
- set_false_path
- set_input_delay
- set_max_delay
- set_min_delay
- set_multicycle_path
- set_output_delay

Send Feedback

report_frames

Print, in textual format, stack frames when current_scope is a process waiting inside subprogram

Syntax

```
report_frames [-quiet] [-verbose]
```

Returns

Returns string

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Simulation

Description

Returns a list of strings of sub-program names, and the calling HDL process in the sub-program call hierarchy for the current HDL process scope, or `current_scope`. The list starts with the HDL process till the most recent sub-program in the hierarchy. Each frame has an associated frame-index. The most recent sub-program is shown at the top, and has an index "0". The symbol (->) is used to indicate the `current_frame`.

By default, the most recently called sub-program frame is the `current_frame`. Other frames can be selected using `current_frame` command. In verbose mode, output gives the source line-file information for each and every call.

> ⭐ **IMPORTANT!** `report_frames` *strictly follows the* `current_scope`*. If the* `current_scope` *is not an HDL process scope waiting inside a sub-program, the command returns an empty list.*

This command returns the name of the design object of the current_instance, or returns nothing when set to the top of current design.

Arguments

`-verbose` - (Optional) Verbose mode. Gives source line-file information for every call. The source file name would include its absolute full path.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

Examples

Example design:

```verilog
module top;
      int i;
      function void f(input int in1);
          automatic int a;
          a = in1 + 7;
          $display($time, " in f :: a %d in1 %d ", a, in1);
      endfunction
      task automatic t(input int in2);
          int b;
          b = in2 + 10;
          $display($time, " in t :: in2 %d b %d ", in2, b);
          #5;
          f(b);      // Case C
          $display($time, " Back in t : after wait and f(%d) ", b);
      endtask
      initial begin                          // "/top/Initial18_0"
          $display($time, " in initial 1 ");
          i = 200;
          t(i);          // Case B
          $display($time, " Back in initial 1 after t(%d) ", i);
      end
      initial begin                          // "/top/Initial25_1"
          $display($time, " in initial 2 ");
          #2;
          f(50);         // Case A
          $display($time, " Back in initial 2 after f(50) ");
      end
  endmodule
```

When simulation is stopped inside function "f" for its call at 'Case C', function "f" is called from task "t" at 'Case C', which itself is called from process "/top/Initial18_0" at 'Case B':

```
> current_scope
   /top/Initial18_0
  1. > report_frames
     -> 0 : f
        1 : t
         2: /top/Initial18_0
  2. > report_frames -verbose
     -> 0 : f @top.v:6
         1 : t @top.v:15
         2 : /top/Initial18_0 @top.v:21
```

See Also

- current_scope
- current_frame

Send Feedback

report_high_fanout_nets

Report high fanout nets

Syntax

```
report_high_fanout_nets [-file <arg>] [-format <arg>] [-append]
    [-ascending] [-timing] [-histogram] [-load_types] [-clock_regions]
    [-slr] [-max_nets <arg>] [-fanout_greater_than <arg>]
    [-fanout_lesser_than <arg>] [-name <arg>] [-cells <args>]
    [-clocks <args>] [-return_string] [-quiet] [-verbose]
```

Returns

Report

Usage

Name	Description
[-file]	Filename to output results to. (send output to console if -file is not used)
[-format]	Specifies how to format the report: text, xml. Default is 'text'. Only applies if -file is used. If xml output is used, -append is not allowed. Default: text
[-append]	Append to existing file
[-ascending]	Report nets in ascending order
[-timing]	Report worst slack and worst delay values on nets
[-histogram]	Report histogram for high fanout nets
[-load_types]	Report load details
[-clock_regions]	Report clock region wise load distribution
[-slr]	Report SLR wise load distribution
[-max_nets]	Number of nets for which report is to be generated Default: 10
[-fanout_greater_than]	Report nets that have fanout greater than the specified integer, default 0 Default: 0
[-fanout_lesser_than]	Report nets that have fanout less than the specified integer, default INT_MAX Default: INT_MAX
[-name]	Output the results to GUI panel with this name
[-cells]	Report the nets of the specified cells
[-clocks]	Report the nets of the specified clocks
[-return_string]	return report as string
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Report](#), [Timing](#)

Send Feedback

Description

Report the fanout of nets in the design, starting with the highest fanout nets, and working down. Options allow you to control various aspects of the report.

This command can be run on an implemented design, or on the synthesized netlist. However, the results will be more complete on the implemented design.

The command returns the fanout report of nets in the design, or returns an error if it fails.

Arguments

`-file <arg>` - (Optional) Write the report into the specified file. The specified file will be overwritten if one already exists, unless `-append` is also specified.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-format [ text | xml ]` - (Optional) The default format of the output report is text. You can also output an XML report. XML output is only valid when `-file` is specified, and cannot be used with `-append`.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-ascending` - (Optional) Report nets in ascending order.

`-timing` - (Optional) Add timing data to the report to display the worst slack (WNS) and worst delay on high fanout nets.

Note: The `-timing` option will slow the run time of `report_high_fanout_nets`, and is not supported for use with `-histogram`.

`-histogram` - (Optional) Format the report as a histogram showing the number of fanouts, and the number and percentage of nets with that many fanouts in the design.

Note: This option cannot be used with the following options: `-ascending`, `-timing`, `-load_types`, `-clock_regions`, `-slr`, `-max_nets`, `-fanout_greater_than`, and `-fanout_lesser_than`.

`-load_types` - (Optional) Reports the various load types on the net sorted in two different ways: by load types (data/clock/set/reset...) and by device resource at which loads are placed (Slices/IO...). When report_high_fanout_nets is run on the unplaced synthesized design only the load type is reported.

`-clock_regions` - (Optional) Report the load distribution across clock regions. This option can only be used after placement, and reports the clock regions the various loads on the net are located in.

`-slr` - (Optional) Report the load distribution across SLRs. This option can only be used after placement, and reports the SLRs that various loads on the net are located in, after placement.

Note: The `-load_types`, `-slr`, and `-clock_regions` options cannot be used together.

`-max_nets <arg>` - (Optional) Number of nets to report. Default: 10

`-fanout_greater_than <arg>` - (Optional) Report only nets that have fanout greater than the specified limit. The value can be specified as an integer, with a default value of 0.

`-fanout_lesser_than <arg>` - (Optional) Report only nets that have fanout less than the specified limit. The value can be specified as an integer, with a default value of INT_MAX.

`-name <arg>` - (Optional) The name of the High Fanout Nets Report view to display in the Vivado IDE when run in GUI mode. If the name has already been used in an open Report view, that view will be closed and a new report opened.

`-cells <args>` - (Optional) Report the nets attached to the specified cell instances in the design. Cells can be passed by name, or as cell objects returned by the `get_cells` command.

`-clocks <args>` - (Optional) Report the nets of the specified clocks. The clocks can be passed by name, or as clock objects returned by the `get_clocks` command.

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example reports the fanout of nets for the specified clocks:

```
report_high_fanout_nets -clocks [get_clocks gt*]
```

The following example reports the nets with fanouts less than 10 loads, and returns the results to a string stored as a Tcl variable:

```
set myRep [report_high_fanout_nets -fanout_lesser_than 10 -return_string]
```

See Also

- get_cells

report_hw_axi_txn

Report formatted hardware AXI Transaction data

Syntax

```
report_hw_axi_txn [-w <arg>] [-t <arg>] [-quiet] [-verbose]
    <hw_axi_txns>...
```

Usage

Name	Description
[-w]	Output data bytes per output line. Default: 8
[-t]	d[SIZE] signed decimal, SIZE bytes per integer, b[SIZE] binary, SIZE bytes per integer, o[SIZE] octal, SIZE bytes per integer, u[SIZE] unsigned decimal, SIZE bytes per integer, x[SIZE] hexadecimal, SIZE bytes per integer Default: x4 (4-bytes in hex)
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hw_axi_txns>	hardware AXI Transaction object to report

Categories

Hardware

Description

Report the results of the specified AXI transactions on the JTAG to AXI Master, hw_axi.

You can use this command after creating `hw_axi_txn` objects on existing hw_axi objects, and then running the hw_axi to exercise the defined transaction.

The JTAG to AXI Master core can only be controlled using Tcl commands. You can issue AXI read and write transactions using the `create_hw_axi_txns` command. However, before issuing these commands, it is important to reset the JTAG to AXI Master core using the `reset_hw_axi` command.

This command reports the transaction data in the specified format, or returns an error if it fails.

Arguments

`-w <arg>` - (Optional) The number of data bytes per output line. The default is 8 bytes per line.

`-t <arg>` - (Optional) Specify the form and size of the output data. The accepted values for formatting the transaction data are:

- d[SIZE] signed decimal, SIZE bytes per integer

- b[SIZE] binary, SIZE bytes per integer

- o[SIZE] octal, SIZE bytes per integer

- u[SIZE] unsigned decimal, SIZE bytes per integer

- x[SIZE] hexadecimal, SIZE bytes per integer

- The default output format is x4, or 4-bytes in hex.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_axi_txns> - (Required) The hw_axi_txn objects to report. The hw_axi_txn must be specified as an object returned by the `get_hw_axi_txns` command.

Example

This example resets the hw_axi core to a known state from which to begin running AXI transactions; then creates an AXI transaction associated with the hw_axi to read the contents of the first four locations of the AXI core; then runs the hw_axi to process the transactions associated with the core; and finally reports the data read from the transaction:

```
reset_hw_axi [get_hw_axis hw_axi_1]
create_hw_axi_txn read_txn [get_hw_axis hw_axi_1] -type READ \
    -address 00000000 -size 32 -len 4 -cache 3 -id 0
run_hw_axi [get_hw_axi_txns [get_hw_axis]]
report_hw_axi_txn [get_hw_axi_txns read_txn]
```

This example creates AXI read and write transactions, runs the hw_axi, and reports on the results:

```
create_hw_axi_txn wr_txn [lindex [get_hw_axis] 0] -address 80000000 \
-data {11112222 33334444 55556666 77778888} -len 4 -type write
create_hw_axi_txn rd_txn [lindex [get_hw_axis] 0] -address 80000000 \
-len 4 -type read
run_hw_axi [get_hw_axi_txns wr_txn]
set wr_report [report_hw_axi_txn wr_txn -w 32]
puts $wr_report
run_hw_axi [get_hw_axi_txns rd_txn]
set rd_report [report_hw_axi_txn rd_txn -w 32]
puts $rd_report
close_hw_target;
disconnect_hw_server;
```

Send Feedback

See Also

- create_hw_axi_txn
- delete_hw_axi_txn
- get_hw_axis
- get_hw_axi_txns
- refresh_hw_axi
- reset_hw_axi

report_hw_ddrmc

Formatted report on Versal integrated and soft Memory Controllers' (DDRMCs) memory configurations, calibration status, stages, and window margins data.

Syntax

```
report_hw_ddrmc [-file <arg>] [-append] [-return_string] [-quiet]
    [-verbose] <hw_objects>
```

Usage

Name	Description
[-file]	file name (including full path) to output the report results to
[-append]	set this option to append the report results to a file
[-return_string]	set this option to have report results returned as a string
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hw_objects>	hardware DDRMC objects

Categories

Report, Hardware

Description

Report formatted information on memory IP hardware configuration, calibration, and margin. Does not include the graphical margin scan plots that are available within the Vivado logic analyzer, or Vivado Lab Edition.

In the Vivado tools, Versal memory controllers implemented into a design are associated with hw_ddrmc objects. These hw_ddrmc objects let you verify the calibration, read, and write window margins in your memory interface design. You can use the hardware manager GUI to check the calibration status, verify the read margin for both rising and falling edges of the clock, and write margin for both simple and complex patterns. You can also use an ILA and VIO core to verify the data integrity for the read and write operations.

This command returns the reported data, or returns an error if it fails.

Arguments

`-file <arg>` - (Optional) Write the report into the specified file.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_objects> - (Optional) Inputs can only be an hw_ddrmc objects, as returned by the `get_hw_ddrmcs` command.

Examples

The following example generates the report on the hw_ddrmc objects and outputs them to the text file specified:

```
report_hw_ddrmc -file C:/Data/ddrmc_report.txt [get_hw_ddrmcs]
```

See Also

- current_hw_device
- current_hw_target
- get_hw_ddrmcs
- refresh_hw_ddrmc

Send Feedback

report_hw_mig

Report formatted hardware MIG calibration status and margin data

Syntax

```
report_hw_mig [-file <arg>] [-append] [-return_string] [-quiet] [-verbose]
    <hw_objects>
```

Usage

Name	Description
[-file]	file name (including full path) to output the report results to
[-append]	set this option to append the report results to a file
[-return_string]	set this option to have report results return as a string
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hw_objects>	hardware mig objects

Categories

Report, Hardware

Description

Report formatted information on memory IP hardware configuration, calibration, and margin. Does not include the graphical margin scan plots that are available within the Vivado logic analyzer, or Vivado Lab Edition.

In the Vivado tools, memory controllers implemented into a design are associated with hw_mig objects. These hw_mig objects let you verify the calibration, read, and write window margins in your memory interface design. You can use the hardware manager GUI to check the calibration status, verify the read margin for both rising and falling edges of the clock, and write margin for both simple and complex patterns, or DQS. You can also use an ILA core to verify the data integrity for the read and write operations.

This command returns the reported data, or returns an error if it fails.

Arguments

`-file <arg>` - (Optional) Write the report into the specified file.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_objects> - (Optional) Inputs can be any hw_mig, hw_device, hw_target, or hw_server object.

Note: The objects must be specified using the appropriate `get_hw_xxx` command, for instance `get_hw_migs`, rather than specified by name.

Examples

The following example generates the report on the hw_mig objects and outputs to the text file specified:

```
report_hw_mig -file C:/Data/hw_mig_report.txt [get_hw_migs]
```

See Also

- commit_hw_mig
- current_hw_device
- get_hw_migs
- implement_mig_cores
- refresh_hw_mig
- set_property

report_hw_pcie

Formatted report for PCIe core.

Syntax

```
report_hw_pcie [-file <arg>] [-append] [-return_string] [-quiet] [-verbose]
    <hw_pcie>
```

Usage

Name	Description
[-file]	File name (including full path) to output the report results to
[-append]	Append the report results to a file
[-return_string]	Return report results as a string
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hw_pcie>	Hardware PCIe object

Categories

[Hardware](#)

Description

Generate a report for the Peripheral Component Interconnect Express (PCIe) debug core object, hw_pcie, defined on the current hardware device.

The customizable LogiCORE IP PCIe core for Xilinx ACAPs is designed for evaluating and monitoring the PCIe Link Training and Status State Machine (LTSSM) running on the Gigabit Transceivers (GTs). In the Vivado Hardware Manager, Versal PCIe soft cores implemented in the design, are represented as hw_pcie objects. You can use these PCIe debug cores to solve a range of debug and validation problems; from viewing the PCIe link info to the LTSSM state transition diagram.

This command generates a report with general information about the PCIe core, the LTSSM state visitation status and the trace data for the LTSSM transition graph.

Arguments

`-file <arg>` - (Optional) Write the report into the specified file.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

Send Feedback

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the -file option.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<hw_pcie>` - (Required) List of HW PCIe objects. If not provided, report is generated for all available PCIe cores.

Examples

The following example generates a report for the PCIe core at index 0

```
report_hw_pcie [lindex [get_hw_pcies] 0]
```

The following example generates the report for the hw_pcie objects and outputs them to the text file specified

```
report_hw_pcie -file C:/data/pcie_report.txt [get_hw_pcies]
```

See Also

- get_hw_pcies
- reset_hw_pcie
- refresh_hw_pcie

report_hw_softmc

Formatted report on Versal soft Memory Controller cores' (Soft MCs) memory configurations, calibration status, stages, and window margins data.

Syntax

```
report_hw_softmc [-file <arg>] [-append] [-return_string] [-quiet]
    [-verbose] <hw_objects>
```

Usage

Name	Description
[-file]	file name (including full path) to output the report results to
[-append]	set this option to append the report results to a file
[-return_string]	set this option to have report results returned as a string
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hw_objects>	hardware Soft MC objects

Categories

Report, Hardware

Description

Report formatted information on memory IP hardware configuration, calibration, and margin. It does not include the graphical margin scan plots that are available within the Vivado logic analyzer or Vivado Lab Edition.

In the Vivado tools, Versal soft memory controllers implemented into a design are associated with `hw_softmc` objects. These `hw_softmc` objects let you verify the calibration, read, and write window margins in your memory interface design. You can use the hardware manager GUI to check the calibration status, verify the read margin for both rising and falling edges of the clock, and write margin for both simple and complex patterns. You can also use an ILA and VIO core to verify the data integrity for the read and write operations.

> ⭐ **IMPORTANT!** *This command returns the reported data, or returns an error if it fails.*

Arguments

`-file <arg>` - (Optional) Write the report into the specified file.

Send Feedback

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The -append option can only be used with the -file option.

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<hw_objects>` - (Required) Inputs can only be `hw_softmc` objects, as returned by the `get_hw_softmcs` command.

Examples

The following example generates the report on the `hw_softmc` objects and outputs them to the text file specified:

```
report_hw_softmc -file C:/Data/softmc_report.txt [get_hw_softmcs]
```

See also

- [current_hw_device](#)
- [current_hw_target](#)
- [get_hw_softmcs](#)
- [refresh_hw_softmc](#)

report_hw_targets

Report properties on hardware objects

Syntax

```
report_hw_targets [-quiet] [-verbose]
```

Returns

Hardware objects

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Hardware

Description

This command returns properties related to the configuration of all hw_targets on the current_hw_server object. The information reported by this command includes:

- Server Property Information: The properties of the current_hw_server, including HOST and PORT.

- Target Property Information: Reported for each target on the hw_server, including NAME, FREQUENCY, DEVICE_COUNT, and SVF.

- Device Property Information: Reported for each device on a specific hw_target, including PART, ID CODE, IR LENGTH, MASK, PROGRAMMING and PROBES FILE.

This command returns the requested information if successful, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

Send Feedback

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example reports the property information for all targets on the connected hw_server:

```
report_hw_targets
```

See Also

- connect_hw_server
- create_hw_target
- get_hw_targets
- open_hw_target

Send Feedback

report_incremental_reuse

Compute achievable incremental reuse for the given design-checkpoint and report

Syntax

```
report_incremental_reuse [-file <arg>] [-append] [-cells <args>]
    [-hierarchical] [-hierarchical_depth <arg>] [-return_string] [-quiet]
    [-verbose]
```

Usage

Name	Description
[-file]	Filename to output results to. (send output to console if -file is not used)
[-append]	Append to existing file
[-cells]	Report incremental reuse of given list of cells
[-hierarchical]	Generates text-based hierarchical incremental reuse report.
[-hierarchical_depth]	Specifies the depth level for textual hierachical incremental reuse report Default: 0
[-return_string]	Return report as string
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Report](#)

Description

For use with the incremental implementation flow, this command reports on the amount of design overlap between the current design and an incremental checkpoint loaded using the `read_checkpoint -incremental` command.

This report analyzes the loaded incremental checkpoint against the current design to see if the two are sufficiently correlated to drive incremental placement and routing. A low correlation between the current design and the checkpoint should discourage using the checkpoint as a basis for incremental place and route. Refer to the *Vivado Design Suite User Guide: Implementation* (UG904) for more information on incremental place and route.

If there is a low correlation of reuse between the current design and the loaded incremental checkpoint, you will need to restore the original design using `open_run` or `read_checkpoint`. Alternatively, you can overload the incremental checkpoint in the current design by issuing the `read_checkpoint -incremental` command again to specify a new incremental checkpoint.

Send Feedback

For Partial Reconfiguration (PR) designs, the % of cells matching in Reconfigurable Modules (RM) are reported, the % of cells reused in RMs, and the % of cells fixed in RMs are also reported. There is also a Reconfigurable Modules Summary table added to the report.

Arguments

`-file <arg>` - (Optional) Write the report into the specified file. The specified file will be overwritten if one already exists, unless `-append` is also specified. By default, the report will be written to the Tcl console.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-cells <args>` - (Optional) Specifies the cells to use from the DCP file.

`-hierarchical` - (Optional) Generate a text-based hierarchical incremental reuse report.

`-hierarchical_depth <arg>` - (Optional) Specifies the depth level for the text-based hierarchical incremental reuse report. The default is 0.

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example loads an incremental checkpoint into the current design, and then reports the correlation of the loaded incremental checkpoint to the current design:

```
read_checkpoint -incremental C:/Data/reuse_checkpoint1.dcp
report_incremental_reuse
```

Send Feedback

See Also

- open_run
- place_design
- read_checkpoint
- route_design

Send Feedback

report_io

Display information about all the IO sites on the device

Syntax

```
report_io [-file <arg>] [-name <arg>] [-append] [-format <arg>]
    [-return_string] [-quiet] [-verbose]
```

Returns

Report

Usage

Name	Description
[-file]	Filename to output results to. Send output to console if -file is not used.
[-name]	Output the results to GUI panel with this name
[-append]	Append to existing file
[-format]	Specifies how to format the report: text, xml. Default is 'text'. Only applies if -file is used. If xml output is used, -append is not allowed. Default: text
[-return_string]	return report as string
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Report

Description

Report details of the IO banks of the current design. Details include device specific information such as target part, package, and speed grade, and also provides information related to each pin on the device.

This command returns the requested report, or returns an error if it fails.

Arguments

`-file <arg>` - (Optional) Write the report into the specified file.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

Send Feedback

`-name <arg>` - (Optional) The name to assign to the reported results when run in GUI mode.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option. It cannot be specified with `-format xml`.

`-format [ xml | text ]` - (Optional) Specifies the format of the output as either XML, or an ASCII text file. The default output is text.

Note: The format applies when `-file` is specified, but is otherwise ignored.

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example reports the IO blocks of the current design:

```
report_io
```

See Also

- report_route_status

report_ip_status

Report on the status of the IP instances in the project.

Syntax

```
report_ip_status [-name <arg>] [-file <arg>] [-append] [-return_string]
    [-license_status] [-resource_data] [-quiet] [-verbose]
```

Returns

True for success.

Usage

Name	Description
[-name]	Output the results to GUI panel with this name Values: The name of the GUI dialog
[-file]	Filename to output results to (send output to console if -file is not used) Values: The report filename
[-append]	Append to existing file
[-return_string]	Return report as string
[-license_status]	Report the license status of the generated outputs for each IP
[-resource_data]	Report the resource data usage for each IP instance
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

IPFlow

Description

This command examines the IP cores in the current project, and reports the state of the IP with regard to the latest IP catalog. The following information is included in the IP Status report:

- Instance Name - The name of the IP core in the current project.

- IP Status - A description of the state of the IP in the current project.

- Recommendation - A recommended action based on the status.

- Lock Status - An explanation of the lock status of the IP in the current project.

- Change Log - A reference to the change log for the IP update in the catalog. This will provide a description of the changes in the latest IP.

Send Feedback

- IP Name - The name of the IP core in the catalog.

- IP Version - The version of the IP in use in the current project.

- New Version - The latest version of the IP in the catalog.

- New license - The license status for the new IP version.

- Original Part - The original part associated with the IP in the catalog.

IP cores that are out of date, or locked, may need to be upgraded and the output products regenerated. Refer to the *Vivado Design Suite User Guide: Designing with IP (UG896)* for more information.

The `report_ip_status` command checks the available licenses on the local machine, or on the license server, for all IP cores in the current project. If a license can be found, the license information is printed. If the license cannot be found, this information is also printed.

This command returns the IP status report, or returns an error if it fails.

This command can also be used to report the resource data usage for each IP instance. This data is obtained from the synthesis results.

- To see the resource data usage in the Tcl console window, type "`report_ip_status -resource_data`".

- To send the report to the return string, type "`report_ip_status -resource data -return_string`".

- To send the report to a file, type "`report_ip_data -resource data -file <filename>`".

Arguments

`-name <arg>` - (Optional) The name to assign to the results when run in GUI mode.

`-file <arg>` - (Optional) Write the report into the specified file. The specified file will be overwritten if one already exists, unless `-append` is also specified.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

Send Feedback

`-license_status` - (Optional) Limits the report to display only the various requirements and state of the licenses for IP used in the design.

`-resource_data` - (Optional) Reports the resource data usage for each IP instance in the design.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example reports the IP status to the specified file, appending the results if the file already exists:

```
report_ip_status -file C:/Data/reports/ip_status.txt -append
```

See Also

- get_ips
- import_ip
- read_ip
- upgrade_ip

report_methodology

Methodology Checks

Syntax

```
report_methodology [-name <arg>] [-cells <args>] [-checks <args>]
    [-file <arg>] [-rpx <arg>] [-append] [-waived] [-no_waivers]
    [-slack_lesser_than <arg>] [-return_string] [-merge_exceptions <arg>]
    [-quiet] [-verbose]
```

Usage

Name	Description
[-name]	Output the results to GUI panel with this name
[-cells]	Run report_methodology on the specified cell(s).
[-checks]	Report Methodology checks (see get_methodology_checks for available checks)
[-file]	Filename to output results to. (send output to console if -file is not used)
[-rpx]	Report filename for persisted results.
[-append]	Append the results to file, do not overwrite the results file
[-waived]	Output result is Waived checks
[-no_waivers]	Disable waivers for checks
[-slack_lesser_than]	Set SYNTH rules Slack Threshold value in 'ns' (float) Default: 2.0
[-return_string]	return report as string
[-merge_exceptions]	Allow/Prevent the timing engine from merging timing exceptions. Valid values are auto/true/false Default: auto
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Methodology, Report, Timing

Description

Check the current design against a specified set of methodology checks and report any errors or violations that are found.

Methodology checks are a special class of design rule checks (DRC) that are accessible through this separate Tcl command. The methodology checks are a necessary part of the design flow, and should be considered mandatory after implementation and prior to generating the bitstream.

Send Feedback

> 💡 **TIP:** *Other than their availability through the separate* `report_methodology` *command, the checks are standard design rule checks in every other way.*

The `report_methodology` command requires an open design to check the design rules against. The command returns a report with the results of violations found by the design rule checks. Violations are returned as Vivado objects that can be listed with the `get_methodology_violations` command, and are associated with cells, pins, ports, nets, and sites in the current design. You can get the cells, nets, and other design objects that are associated with methodology violation objects, using the `-of_objects` option of the `get_cells` command for instance.

The `report_methodology` command runs the methodology rule deck, or you can use the `-checks` option to specify the set of checks to run. Methodology checks can also be enabled or disabled in the default rule decks using the `IS_ENABLED` property on the rule check object:

```
set_property IS_ENABLED FALSE [get_methodology_checks PDRC-190]
```

If a rule `IS_ENABLED` false, the rule will not be run by the `report_methodology` command.

> 💡 **TIP:** *You can reset the properties of a methodology rule to the factory default settings using the* `reset_methodology_check` *command.*

You can reset the current results of the `report_methodology` command, clearing any found violations, using the `reset_methodology` command.

Arguments

`-name <arg>` - (Optional) The name to assign to the results when run in GUI mode.

`-cells <arg>` - (Optional) Specify the cells to use when running `report_methodology`. Cells should be hierarchical modules of the design. Use this option to specify a cell other than the top cell for analysis. Cells can be specified by name, or as objects returned by the `get_cells` command.

`-checks <args>` - (Optional) A list of rule checks to run the methodology report against. All specified rules will be checked against the current design. Rules are listed by their group name or full key. Using the `-checks` option creates a temporary user-defined rule deck, with the specified design rule checks, and uses the temporary rule deck for the run.

`-file <arg>` - (Optional) Write the methodology report into the specified file. The specified file will be overwritten if one already exists, unless `-append` is also specified.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-rpx <arg>` - (Optional) Specify the file name and path of an Xilinx report file (RPX) to write. This is different from writing the report results to a file using the `-file` argument. The RPX file is an interactive report that contains all the report information and can be reloaded into memory in the Vivado Design Suite using the `open_report` command. You should add a `.rpx` file extension to the specified file name, as the Vivado tool will not automatically assign a file extension.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-waived` - (Optional) Causes only the methodology checks waived by the `create_waiver` command to be run and reported. This returns the actual violation rather than the definition of the waiver, which can be reported by the `report_waivers` command.

`-no_waivers` - (Optional) Ignore the waivers defined by the `create_waivers` command and report all methodology violations.

`-slack_lesser_than <arg>` - (Optional) Only report SYNTH methodology violations when the calculated slack value on a path is less than the specified value.

`-merge_exceptions [ auto | true | false ]` - (Optional) Allow or prevent the timing engine from merging the timing exceptions. Some methodology violations include the constraint position number inside the message. This information can only be reported when the exceptions are not merged, but at the expense of runtime and memory. In `auto` mode, when the tool estimates that the complexity of the timing constraints could result in a significant higher runtime, the exceptions are not merged. The user can override the default behavior by forcing the exceptions to be always merged (`true`) or never merged (`false`).

Note: The valid values are `true`, `false` or `auto` with a default setting of auto.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

Run the default methodology checks against the current design and write the results to the specified file.

```
report_methodology -file C:/Data/methodology_Rpt1.txt -append
```

Note: The `-append` option adds the result to the specified file.

See Also

- get_methodology_checks
- get_methodology_violations
- open_report
- report_drc
- reset_methodology

report_objects

Print details of the given hdl objects (variable, signal, wire, or reg)

Syntax

```
report_objects [-quiet] [-verbose] [<hdl_objects>...]
```

Returns

Print name, type, data_type of the HDL objects on console in textual format.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<hdl_objects>]	The hdl_objects to report. Default is report_objects [get_objects *]

Categories

Simulation

Description

The `report_objects` command reports the type, name, and language of the specified HDL objects to the Tcl Console or Tcl shell. You must have an open simulation to use this command.

This command returns a brief description of the specified objects. Use the `describe` command to return more detailed information.

HDL objects include HDL signals, variables, or constants as defined in the Verilog or VHDL test bench and source files. An HDL signal includes Verilog wire or reg entities, and VHDL signals. Examples of HDL variables include Verilog real, realtime, time, and event. HDL constants include Verilog parameters and localparams, and VHDL generic and constants.

The command returns the HDL object type, the name, and the code type (Verilog/VHDL) for each object, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hdl_objects> - (Optional) Specifies the objects to report. The default command reports all objects found in the current scope of the simulation, or `current_scope`.

Note: Objects can be specified by name, or returned as objects by the `get_objects` command.

Examples

The following example shows how the specified objects reported depend upon the current scope of the simulation:

```
current_scope testbench
/testbench
report_objects [get_objects leds_n]
  Declared: {leds_n[3:0]}                    Verilog
current_scope dut
/testbench/dut
report_objects leds_n
       Out: {LEDS_n[3:0]}                     VHDL
```

This example reports the specified HDL objects of the current simulation scope:

```
report_objects [get_objects GPIO*]
```

See Also

- current_scope
- describe
- get_objects

report_operating_conditions

Get operating conditions values for power estimation

Syntax

```
report_operating_conditions [-voltage <args>] [-voltage_range <args>]
    [-grade] [-process] [-junction_temp] [-ambient_temp] [-thetaja]
    [-thetasa] [-airflow] [-heatsink] [-thetajb] [-board] [-board_temp]
    [-board_layers] [-design_power_budget] [-all] [-file <arg>]
    [-return_string] [-append] [-supply_current_budget <args>] [-quiet]
    [-verbose]
```

Usage

Name	Description
[-voltage]	Gets voltage value. Supported voltage supplies vary by family.
[-voltage_range]	Gets voltage range. Supported voltage supplies vary by family.
[-grade]	Temperature grade. Supported values vary by family.
[-process]	Gets process
[-junction_temp]	Junction Temperature (C): auto|degC
[-ambient_temp]	Ambient Temperature (C): default|degC
[-thetaja]	ThetaJA (C/W): auto|degC/W
[-thetasa]	Gets ThetaSA
[-airflow]	Airflow (LFM): 0 to 750
[-heatsink]	Gets dimensions of heatsink
[-thetajb]	Gets ThetaJB
[-board]	Board type: jedec, small, medium, large, custom
[-board_temp]	Board Temperature degC
[-board_layers]	Board layers: 4to7, 8to11, 12to15, 16+
[-design_power_budget]	Design Power Budget (W)
[-all]	Gets all operating conditions listed in this help message
[-file]	Filename to output results to. (send output to console if -file is not used)
[-return_string]	return operating conditions as string
[-append]	append operating conditions to end of file
[-supply_current_budget]	Reports a list of supply current budget 'name value' pairs. Supported voltage supplies vary by family.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Report

Description

Displays the real-world operating conditions that are used when performing analysis of the design. The reported values of operating conditions can be defined by the `set_operating_conditions` command.

The environmental operating conditions of the device are used for power analysis when running the `report_power` command, but are not used during timing analysis.

Note: By default the report is written to the Tcl console or STD output. However, the results can also be written to a file or returned as a string if desired.

Arguments

`-voltage` - (Optional) Report the list of voltage pairs. Supported voltage supplies vary by family.

`-voltage_range` - (Optional) Report the list of voltage ranges. Supported voltage supplies vary by family.

`-grade` - (Optional) Report the temperature grade of the target device

`-process` - (Optional) Report the manufacturing process characteristics to be assumed.

`-junction_temp` - (Optional) Report the device junction temperature used for modeling

`-ambient_temp` - (Optional) Reports the environment ambient temperature

`-thetaja` - (Optional) Report the Theta-JA thermal resistance used for modeling

`-thetasa` - (Optional) Report the Theta-SA thermal resistance used for modeling

`-airflow` - (Optional) Report the Linear Feet Per Minute (LFM) airflow to be used for modeling.

`-heatsink` - (Optional) Report the heatsink type to be used for modeling.

`-thetajb` - (Optional) Report the Theta-JB thermal resistance used for modeling

`-board` - (Optional) Report the board size to be used for modeling.

`-board_temp` - (Optional) Report the board temperature in degrees Centigrade to be used for modeling.

`-board_layers` - (Optional) Report the number of board layers to be used for modeling

`-design_power_budget` - (Optional) Report the design power budget in Watts. This value is used by the `report_power` command to report the difference between the calculated on-chip power and the design power budget.

`-all` - (Optional) Report the current values of all operating conditions. Use this to avoid having to report each condition separately.

`-file <arg>` - (Optional) Write the report into the specified file. This will overwrite an existing file of the same name, unless the `-append` option is also used.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-supply_current_budget` - (Optional) Report the list of supply name and current budget pairs. Supported voltage supplies vary by family.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

Specify an industrial temperature grade device with an ambient temperature of 75 degrees C and then write those settings to a file on disk.

```
set_operating_conditions -grade industrial -junction_temp 75
report_operating_conditions -grade -junction_temp -return_string -file \
~/conditions.txt
```

Get voltage ranges of VCCINT and VCCAUX for the target device.

```
report_operating_conditions -voltage_range {VCCINT VCCAUX}
```

See Also

• set_operating_conditions

report_param

Get information about all parameters

Syntax

```
report_param [-file <arg>] [-append] [-non_default] [-return_string]
    [-quiet] [-verbose] [<pattern>]
```

Returns

Param report

Usage

Name	Description
[-file]	Filename to output results to. (send output to console if -file is not used)
[-append]	Append the results to file, don't overwrite the results file
[-non_default]	Report only params that are set to a non default value
[-return_string]	Return report as string
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<pattern>]	Display params matching pattern Default: *

Categories

PropertyAndParameter, Report

Description

Gets a list of all user-definable parameters, the current value, and a description of what the parameter configures or controls.

Arguments

-file <arg> - (Optional) Write the report into the specified file. The specified file will be overwritten if one already exists, unless -append is also specified.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

-append - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The -append option can only be used with the -file option.

`-non_default` - (Optional) Report only the parameters that have been changed from their default values.

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<pattern>` - (Optional) Match parameters against the specified pattern. The default pattern is the wildcard '*' which gets all user-definable parameters.

Examples

The following example returns the name, value, and description of all user-definable parameters:

```
report_param
```

The following example returns the name, value, and description of user-definable parameters that match the specified search pattern:

```
report_param *coll*
```

See Also

- get_param
- list_param
- reset_param
- set_param

report_phys_opt

Report details of Physical Synthesis transformations.

Syntax

```
report_phys_opt [-file <arg>] [-append] [-return_string] [-quiet]
    [-verbose]
```

Usage

Name	Description
[-file]	Output file
[-append]	Append the results to file
[-return_string]	return report as string
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Report](#)

Description

Reports the results of the fanout driver replication and load redistribution optimizations performed by the `phys_opt_design` command.

Arguments

`-file <arg>` - (Optional) Write the physical optimization report into the specified file. The specified file will be overwritten if one already exists, unless `-append` is also specified.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-return_string` - (Optional) Directs the output of the report to a Tcl string. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example reports the physical optimizations performed in the current design by the `phys_opt_design` command:

```
report_phys_opt -file C:/Data/physOpt_Report.txt
```

See Also

- phys_opt_design

Send Feedback

report_pipeline_analysis

Perform pipeline register insertion analysis and display report.

Syntax

```
report_pipeline_analysis [-cells <args>] [-verbose] [-clocks <args>]
    [-file <arg>] [-include_paths_to_pipeline] [-append]
    [-max_added_latency <arg>] [-report_loops] [-return_string] [-quiet]
```

Usage

Name	Description
[-cells]	Analyze each of the specified hierarchical cells separately and ignore feedback loops external to the cells.
[-verbose]	Suspend message limits during command execution
[-clocks]	Filter report output to show only the specified clocks
[-file]	Filename to output results to. (send output to console if -file is not used)
[-include_paths_to_pipeline]	Report paths to cut. (only available if -file is used)
[-append]	Append to existing file
[-max_added_latency]	Maximum extra latency that can be inserted into the system (0 = unlimited). Default: 100
[-report_loops]	Report loop information as well
[-return_string]	return report as string
[-quiet]	Ignore command errors

Categories

Tools

Description

This command performs an analysis of a synthesized design, hypothetically inserting pipeline stages in the design and reports the potential frequency (Fmax) increase of each clock domain. The analysis includes a search for loops in the design, which may not be improved by pipelining, and determines if such loops are critical paths in the design.

Returns a table showing the pipeline stages and the Fmax improvement. The report begins with the original design and adds stages of latency (1, 2, ...) until there is no further improvement in Fmax. This reports a theoretical upper limit to the frequency performance of the design.

The analysis is typically run on the un-placed synthesized netlist where the logical netlist structure determines the performance. The report can be run on the top-level design, or on out-of-context (OOC) sub-modules. This report confirms whether the design frequency can be increased, as well as how many pipeline registers must be added to the design to achieve the Fmax improvement.

Note: By default the report is written to the Tcl console or STD output. However, the results can also be written to a file.

Arguments

`-cells` <args> - (Optional) Perform pipeline analysis for the specified hierarchical cells. Specifying multiple cells causes the pipeline analysis report to be generated for each cell. The cells of interest can be specified by name, or returned as an object using the `get_cells` command.

`-verbose <arg>` - (Optional) Specify the level of detail in the returned report. The argument can be specified with an integer value greater than 0.

`-clocks` <args> - (Optional) Specifies the clock domains to analyze when generating the report. If not specified, the timing paths for all clocks are analyzed. This limits results to paths groups involving the specified clock domains.

`-file <arg>` - (Optional) Write the report into the specified file. The specified file will be overwritten if one already exists, unless `-append` is also specified.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-include_paths_to_pipeline` - (Optional) Report recommendations for paths to pipeline in the current design.

> 💡 **TIP:** *The `-file` option must also be specified when this option is used.*

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-max_added_latency <arg>` - (Optional) Specify the additional levels of delay to add through pipeline insertion. The latency is specified as an integer from 1 to 100, representing the maximum number of pipeline stages to consider during the pipeline analysis. The default setting is 0, which directs the tool to insert pipeline delays until there is no further improvement in design performance. The tool analyzes the number of stages up to the specified limit, or until there is not further gain in Fmax.

`-report_loops` - (Optional) Report the slowest path within a sequential feedback loop. These are paths starting from and ending at the same sequential cell, and may have zero, one, or more sequential cells in the feedback path. Sequential loops cannot be pipelined.

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

Examples

The following example returns the name, period, waveform, and sources of the clocks in the current design:

```
report_pipeline_analysis -file C:/Data/FPGA_Design/pipeline_report.txt
```

See Also

- get_cells
- get_clocks
- report_cdc
- report_design_analysis

report_power

Run power estimation and display report

Syntax

```
report_power [-no_propagation] [-hier <arg>] [-hierarchical_depth <arg>]
    [-vid] [-advisory] [-file <arg>] [-name <arg>] [-format <arg>]
    [-xpe <arg>] [-l <arg>] [-return_string] [-append] [-rpx <arg>]
    [-quiet] [-verbose]
```

Usage

Name	Description
[-no_propagation]	Disables the propagation engine to estimate the switching activity of nets.
[-hier]	Hierarchy report style (logic, power, or all) Default: power
[-hierarchical_depth]	Specifies the depth level for textual hierarchical report Default: 4
[-vid]	Voltage ID (VID) of device is used
[-advisory]	Dump power advisory text report
[-file]	Filename to output results to. (send output to console if -file is not used)
[-name]	Output the results to GUI panel with this name
[-format]	Format for the power estimation report: text, xml Default: text
[-xpe]	Output the results to XML file for importing into XPE
[-l]	Maximum number of lines to report in detailed reports (l >= 0) Default: 10
[-return_string]	return report as string
[-append]	append power report to end of file
[-rpx]	Filename to output interactive results to.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Report](), [Power]()

Description

Run power analysis on the current design, and report details of power consumption based on the current operating conditions of the device, and the switching rates of the design. The operating conditions can be set using the `set_operating_conditions` command. The switching activity can be defined using the `set_switching_activity` command.

Send Feedback

Switching activity can also be read in from an SAIF file with the `read_saif` command. The Vivado tool will annotate the design nodes with activity from the SAIF file and estimate power appropriately.

Power analysis requires an open synthesized design, or implemented design.

Note: By default the report is written to the Tcl console or STD output. However, the results can also be written to a file or returned as a string if desired.

Arguments

`-no_propagation` - (Optional) For all undefined nodes power analysis uses a vector-less propagation engine to estimate activity. This argument disables the propagation engine for a faster analysis of the design.

`-hier [ power | logic | all ]` - (Optional) Displays the summary power consumption for each level of design hierarchy (`power`), or the power broken-down for different logic elements of the hierarchy (`logic`), or both the power summary and the different logic elements of the design hierarchy (`all`). The default is `power`.

`-hierarchical_depth <arg>` - (Optional) Specifies the depth of the hierarchy to report in the "By hierarchy" section of the power report. The default depth is 4, which means that it will report the top-level and three hierarchical levels of child instances by default. A depth of 0 will report all levels of hierarchy. This option is only for textual power report and does not impact GUI or rpx reports.

`-vid` - (Optional) Use the Voltage ID bit of the target device. Voltage identification is a form of adaptive voltage scaling (AVS) that enables certain devices in the Virtex®-7 family to be operated at a reduced voltage of 0.9V while delivering the same specified performance of a device operating at the nominal supply voltage of 1.0V. Voltage identification capable devices consume approximately 30% lower worst case (maximum) static power and correspondingly dissipate less heat.

`-advisory` - (Optional) Adds the Advisory table to the Power Report checking the design for abnormal switching activity on control signals. This is the same table produced by the Power Constraints Advisor feature in the Vivado IDE.

`-file <arg>` - (Optional) Write the report into the specified file. The specified file will be overwritten if one already exists, unless `-append` is also specified.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-name <arg>` - (Optional) Specifies the name of the results set to report the results to.

`-format [ text | xml ]` - (Optional) The default format of the output report is text. You can also output an XML report. XML output is only valid when `-file` is specified, and cannot be used with `-append`.

`-xpe <arg>` - (Optional) Output the results to an XML file for importing into the Xilinx® Power Estimator spreadsheet tool. Refer to *Xilinx Power Estimator User Guide* (UG440) for more information.

`-l <arg>` - (Optional) Maximum number of lines to report in the Detailed Reports section. Valid values are greater than or equal to 0.

Note: This options also triggers additional levels of detail in the Detailed Report section that are not reported when `-l` is not specified.

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-rpx <arg>` - (Optional) Specify the file name and path of an Xilinx report file (RPX) to write. This is different from writing the report results to a file using the `-file` argument. The RPX file is an interactive report that contains all the report information and can be reloaded into memory in the Vivado Design Suite using the `open_report` command. You should add a `.rpx` file extension to the specified file name, as the Vivado tool will not automatically assign a file extension.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example performs power analysis, without net propagation, and writes the results to an XML file for use in XPE:

```
report_power -no_propagation -xpe C:/Data/design1.xpe
```

See Also

- read_saif
- set_switching_activity
- set_operating_conditions

Send Feedback

report_power_opt

Report power optimizations

Syntax

```
report_power_opt [-cell <args>] [-file <arg>] [-format <arg>] [-name <arg>]
    [-append] [-return_string] [-quiet] [-verbose]
```

Usage

Name	Description
[-cell]	list of instance names Default: empty
[-file]	output file
[-format]	Specifies how to format the report. Default is 'text', another option is 'xml'. Only applies if -file is used. If xml output is used, -append is not allowed. Default: text
[-name]	Output the results to GUI panel with this name
[-append]	append if existing file. Otherwise overwrite existing file.
[-return_string]	return report as string
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Power

Description

Report power optimizations that have been performed on the design with the `power_opt_design` command.

Note: By default the report is written to the Tcl console or STD output. However, the results can also be written to a file or returned as a string if desired.

Arguments

`-cell` <args> - (Optional) Report power optimization for the specified cell or cell instances. By default, the power optimizations performed on the whole design are reported.

`-file` <arg> - (Optional) Write the report into a file. The specified file will be overwritten if one already exists.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

Send Feedback

`-format [ text | xml ]` - (Optional) Specify the file format of the Power Optimization report when used with `-file`. The default file output is a text format file. The Vivado tool can also write an XML format file to allow for use by third party applications.

Note: When `-format xml` is specified, the `-append` option is not supported.

`-name <arg>` - (Optional) The name to assign to the Power Optimization report when the command is run in GUI mode.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option, and is not compatible with the XML format.

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example reports the power optimizations performed on the current design, writing them to the specified file in an XML format:

```
report_power_opt -format xml -file C:/Data/power_opt.xml
```

See Also

- power_opt_design
- report_power

Chapter 3:

report_pr_configuration_analysis

Report reconfigurable partition analysis across multiple configurations

Syntax

```
report_pr_configuration_analysis [-complexity] [-clocking] [-timing]
    [-cells <args>] [-dcps <args>] [-rent] [-nworst <arg>] [-file <arg>]
    [-quiet] [-verbose]
```

Usage

Name	Description
[-complexity]	Run complexity analysis
[-clocking]	Run clocking analysis
[-timing]	Run boundary net timing analysis
[-cells]	List of reconfigurable cell names
[-dcps]	List of design checkpoints for each reconfigurable cell. The order of dcps must match that of the -cells option.
[-rent]	Compute Rents component as part of complexity analysis. Runtime intensive for large designs.
[-nworst]	Specifies the N worst boundary paths. Default: 10
[-file]	Filename to output results to. (send output to console if -file is not used)
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Report](#)

Description

⭐ **IMPORTANT!** *You must first define the project as a Partial Reconfiguration (PR) project by setting the PR_FLOW property on the project to TRUE, or by using the* **Tools → Enable Partial Reconfiguration** *command.*

Report reconfigurable partition analysis across multiple configurations as defined by `create_pr_configuration`. This report compares each Reconfigurable Module that you select to give you input on your PR design. It examines resource usage, floorplanning, clocking, and timing metrics to help you manage the overall PR design. For more information on this command refer to the *Vivado Design Suite User Guide: Dynamic Function eXchange* (UG909).

When this analysis is done, each RM is examined based on information in the checkpoints provided. While post-synthesis checkpoints can be supplied, the most complete information is not available until after opt_design when all the linking and expansion has been done.

Send Feedback

Arguments

`-complexity` - (Optional) Perform complexity analysis of the design and report the Rent Exponent, Average Fanout, and Primitive Histogram.

> 💡 **TIP:** *The `-complexity` option can be specified with `-cells` to control the analysis of the design or cells.*

`-clocking` - (Optional) This option focuses the report on clock usage and loads for each RM, helping you plan the overall clocking distribution of the design.

`-timing` - (Optional) This option focuses the report on boundary interface timing details, allowing you analyze bottlenecks in and out of RMs.

`-cells` <args> - (Option) Generate the report on the specified list of reconfigurable cells. The details of the report will be based on the specified cells rather than the whole design, or `current_instance`.

`-dcps` <args> - (Optional) List of design checkpoints (DCP) for each reconfigurable cell.

> ⭐ **IMPORTANT!** *The order of DCP files specified must match the order of the `-cells` option.*

`-rent` - (Optional) Adds Rent metrics to the complexity analysis. This is runtime intensive for large designs, which is why it is not included in complexity analysis by default.

`-nworst <arg>` - (Optional) Reports the N worst boundary paths. The default is 10.

`-file <arg>` - (Optional) Write the analysis results into the specified file instead of to the Tcl console. The specified file will be overwritten if one already exists.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example performs complexity analysis of the design:

```
report_pr_configuration_analysis -complexity
```

See Also

- create_partition_def
- create_pr_configuration
- delete_reconfig_modules
- get_partition_defs
- get_reconfig_modules
- set_property

Send Feedback

report_property

Report properties of object

Syntax

```
report_property [-all] [-class <arg>] [-return_string] [-file <arg>]
      [-append] [-regexp] [-quiet] [-verbose] [<object>] [<pattern>]
```

Returns

Property report

Usage

Name	Description
[-all]	Report all properties of object even if not set
[-class]	Object type to query for properties. Not valid with <object>
[-return_string]	Set the result of running report_property in the Tcl interpreter's result variable
[-file]	Filename to output result to. Send output to console if -file is not used
[-append]	Append the results to file, don't overwrite the results file
[-regexp]	Pattern is treated as a regular expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<object>]	Object to query for properties
[<pattern>]	Pattern to match properties against Default: *

Categories

Object, PropertyAndParameter, Report

Description

Gets the property name, property type, and property value for all of the properties on a specified object, or class of objects.

Note: `list_property` also returns a list of all properties on an object, but does not include the property type or value.

You can specify objects for `report_property` using the `get_*` series of commands to get a specific object. You can use the `lindex` command to return a specific object from a list of objects:

```
report_property [lindex [get_cells] 0]
```

Send Feedback

However, if you are looking for the properties on a class of objects, you should use the `-class` option instead of an actual object.

This command returns a report of properties on the object, or returns an error if it fails.

Arguments

`-all` - (Optional) Return all of the properties for an object, even if the property value is not currently defined.

`-class <arg>` - (Optional) Return the properties of the specified class instead of a specific object. The class argument is case sensitive, and most class names are lower case.

Note: `-class` cannot be used together with an `<object>`.

`-return_string` - (Optional) Directs the output to a Tcl string. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

`-file <arg>` - (Optional) Write the report into the specified file. The specified file will be overwritten if one already exists, unless `-append` is also specified.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-regexp` - (Optional) Specifies that the search <patterns> are written as regular expressions. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://perldoc.perl.org/perlre.html for help with regular expression syntax.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<object>` - (Optional) A single object on which to report properties.

Note: If you specify multiple objects you will get an error.

<pattern> - (Optional) Match the available properties on the <object> or -class against the specified search pattern. The <pattern> applies to the property name, and only properties matching the specified pattern will be reported. The default pattern is the wildcard '*' which returns a list of all properties on the specified object.

Note: The search pattern is case sensitive, and most properties are UPPER case.

Examples

The following example returns all properties of the specified object:

```
report_property -all [get_cells cpuEngine]
```

The following example returns the properties of the specified class of objects, rather than an actual object:

```
report_property -class bel
```

The following example returns properties on the current hw_device that match the specified pattern, specified as a regular expression:

```
report_property [current_hw_device] -regexp .*PROG.*
```

To determine which properties are available for the different design objects supported by the tool, you can use multiple report_property commands in sequence. The following example returns all properties of the specified current objects:

```
report_property -all [current_project]
report_property -all [current_fileset]
report_property -all [current_design]
report_property -all [current_run]
```

See Also

- create_property
- current_design
- current_fileset
- current_hw_device
- current_project
- current_run
- get_cells
- get_property
- list_property
- list_property_value
- reset_property
- set_property

Send Feedback

report_pulse_width

Report pulse width check

Syntax

```
report_pulse_width [-file <arg>] [-append] [-name <arg>] [-return_string]
    [-warn_on_violation] [-all_violators] [-significant_digits <arg>]
    [-limit <arg>] [-min_period] [-max_period] [-low_pulse] [-high_pulse]
    [-max_skew] [-clocks <args>] [-no_header] [-cells <args>] [-rpx <arg>]
    [-quiet] [-verbose] [<objects>]
```

Usage

Name	Description
[-file]	Filename to output results to. (send output to console if -file is not used)
[-append]	Append the results to file, don't overwrite the results file
[-name]	Results name in which to store output
[-return_string]	return report as string
[-warn_on_violation]	issue a critical warning when the report contains a timing violation
[-all_violators]	Only report pins/ports where check violations occur
[-significant_digits]	Number of digits to display: Range: 0 to 3 Default: 3
[-limit]	Number of checks of a particular type to report per clock: Default is 1 Default: 1
[-min_period]	Only report min period checks
[-max_period]	Only report max period checks
[-low_pulse]	Only report min low pulse width checks
[-high_pulse]	Only report min high pulse width checks
[-max_skew]	Only report max skew checks
[-clocks]	List of clocks for which to report min pulse width/min period checks
[-no_header]	Do not generate a report header
[-cells]	run report_pulse_width on the specified cell(s)
[-rpx]	Filename to output interactive results to.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<objects>]	List of objects to check min pulse width with

Categories

[Report](#), [Timing](#)

Send Feedback

Description

Reports the pulse width of the specified clock signals in the clock network and upon reaching the flip-flop. This command also performs high pulse width checking, using maximum delay for the rising edge and minimum delay for the falling edge of the clock. Performs low pulse width checking using minimum delay for the rising edge, and maximum delay for the falling edge. This results in a worst case analysis for the current Synthesis or Implemented Design because it assumes worst-case delays for both rising and falling edges. This command also reports the maximum skew, or maximum timing separation allowed between clock signals.

The report includes minimum pulse width, maximum pulse width, low pulse width, high pulse width, and max skew checks by default. However, selecting a specific check will disable the other checks unless they are also specified.

The default report is returned to the standard output, but can be redirected to a file, or to a Tcl string variable for further processing. The report is returned to the standard output by default, unless the `-file`, `-return_string`, or `-name` arguments are specified.

Arguments

`-file <arg>` - (Optional) Write the report into the specified file. If the specified file already exists, it will be overwritten by the new report, unless the `-append` option is also specified.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-name <arg>` - (Optional) Specifies the name of the results set for the GUI. Pulse Width reports in the GUI can be deleted by the `delete_timing_results` command.

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-warn_on_violation` - (Optional) Specify that a Critical Warning will be generated by the Vivado Design Suite when the report contains a violation.

`-all_violators` - (Optional) Report only the `<objects>` where violations are found.

`-significant_digits <arg>` - (Optional) The number of significant digits in the output results. The valid range is 0 to 3. The default setting is 2 significant digits.

`-limit <arg>` - (Optional) The number of checks of a particular type to report per clock. This is a value >= 1, and the default is 1.

`-min_period` - (Optional) Report minimum period checks.

`-max_period` - (Optional) Report maximum period checks.

`-low_pulse` - (Optional) Report minimum low pulse width checks.

`-high_pulse` - (Optional) Report minimum high pulse width checks.

`-max_skew` - (Optional) Check the skew constraints between two clock pins.

Note: The default of the `report_pulse_width` command is to report `min_period`, `max_period`, `low_pulse`, `high_pulse`, and `max_skew`. Specifying one or more of these options configures the command to only report the specified checks.

`-clocks <arg>` - (Optional) Clocks to report pulse width and period checks. All clocks are checked if the `-clocks` option is not specified.

`-no_header` - (Optional) Eliminate the report header from the results. This can be especially useful when returning the results as a string with `-return_string`.

`-cells <arg>` - (Option) Generate the report for the specified hierarchical cells. The details of the report will be based on the specified cells rather than the whole design.

`-rpx <arg>` - (Optional) Specify the file name and path of an Xilinx report file (RPX) to write. This is different from writing the report results to a file using the `-file` argument. The RPX file is an interactive report that contains all the report information and can be reloaded into memory in the Vivado Design Suite using the `open_report` command. You should add a `.rpx` file extension to the specified file name, as the Vivado tool will not automatically assign a file extension.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<objects>` - (Optional) The pin objects to report the pulse width from. All pins are checked if no `<objects>` are specified.

Examples

The following example performs the minimum period and low pulse width check, returning the results to a named results set in the GUI:

```
report_pulse_width -min_period -low_pulse -name timing_1
```

See Also

- delete_timing_results

report_qor_assessment

Feasibility Checks

Syntax

```
report_qor_assessment [-file <arg>] [-csv <arg>] [-append]
    [-exclude_methodology_checks] [-max_paths <arg>] [-return_string]
    [-quiet] [-verbose]
```

Usage

Name	Description
[-file]	Filename to output results to. (send output to console if -file is not used)
[-csv]	Write detailed tables to the CSV filename specified
[-append]	Append the results to file, do not overwrite the results file
[-exclude_methodology_checks]	Methodology checks are skipped
[-max_paths]	Number of paths to consider for suggestion analysis Default: 100
[-return_string]	return report as string
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Feasibility](), [Report](), [Timing]()

Description

This command look for trouble spots in the design and assesses the likelihood of a design meeting design goals. This command requires an open elaborated, synthesized or implemented design for analysis.

The Report QoR Assessment covers multiple categories:

- Design Methodology

- Synthesis

- Implementation

- Design Hierarchy

- Partial Reconfiguration

- Floorplanning

The `report_qor_assessment` command includes a subset of `report_methodology` checks identifying bad practices that can lead to an expected problem. Yet, `report_qor_assessment` is also more comprehensive than `report_methodology`, because it includes other checks that are not necessarily bad practices but may have low success due to the structure, style, size, or complexity of the current design.

A key feature of Report QOR Assessment is the ability to predict certain conditions that lead to congestion and performance degradation. This command can be run at any stage of the implementation flow after synthesis. The `report_qor_assessment` command will return a score from 1 - 5, where 1 is a failing grade, and 5 means the design will likely pass implementation and meet timing.

For violations found by Report QOR Assessment, the Report QOR Suggestions command will make recommendations on how to avoid issues, or modify the design to improve results.

Arguments

`-file <arg>` - (Optional) Write the QOR Assessment report into the specified file. The specified file will be overwritten if one already exists, unless `-append` is also specified.

`-csv <arg>` - (Optional) Write a CSV file that shows which timing paths are contributing to the RQA score. Note, not all elements of the RQA score are generated from examining timing paths. The specified file will be overwritten if one already exists.

Note: If the path is not specified as part of the file name for either `-file` or `-csv`, the file will be written into the current working directory that can be identified using the `[pwd]` Tcl command.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-exclude_methodology_checks` - (Optional) By default, methodology checks are included in the report. This switch prevents methodology checks being reported and reduces the time taken to execute the command.

`-report_all_suggestions` - (Optional) When running `report_qor_assessment` followed by `report_qor_suggestions`, this option should match in both commands to reduce the compile time.

`-max_paths <arg>` - (Optional) Specify the number of critical paths to analyze. The default is the 100 worst timing paths.

`-return_string` - (Optional) Direct the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Send Feedback

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

Report suggestions after analyzing the worst 10 paths.

```
report_qor_assessment -max_paths 10
```

See Also

- report_design_analysis
- report_qor_suggestions

report_qor_suggestions

Recommend QoR Suggestions

Syntax

```
report_qor_suggestions [-file <arg>] [-csv <arg>] [-name <arg>] [-append]
    [-return_string] [-max_strategies <arg>] [-max_paths <arg>] [-no_split]
    [-report_all_suggestions] [-cells <args>] [-of_objects <args>] [-quiet]
    [-verbose]
```

Usage

Name	Description
[-file]	Filename to output results to. (send output to console if -file is not used)
[-csv]	Write detailed tables to the CSV filename specified
[-name]	Output the results to GUI panel with this name
[-append]	Append the results to file, don't overwrite the results file
[-return_string]	Return report as string
[-max_strategies]	Number of strategies to suggest Default: 3
[-max_paths]	Number of paths to consider for suggestion analysis Default: 100
[-no_split]	Report without spliting the lines in tables
[-report_all_suggestions]	Report all suggestions
[-cells]	Report QOR suggestions for a given cell
[-of_objects]	List of QoR suggestion objects
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Feasibility](), [Report](), [Timing](), [Object]()

Description

Report design and tool option recommendations related to improving the quality of results (QoR). The report looks at timing constraints, netlist characteristics, failing timing paths, and congestion information to determine suggestions that can enhance the QoR. The report can be generated after synthesis, or after any implementation step, but requires a design to be open.

Send Feedback

The `report_qor_suggestions` command creates QoR suggestion objects related to the suggestions identified. These suggestion objects can be enabled for use by the Vivado tools to improve the quality of synthesis and implementation results. Some of these suggestions can be automatically applied, and some may require more manual intervention to implement design changes, or write Tcl design constraints.

Suggestion objects can be obtained using the `get_qor_suggestions` command. The QoR objects have properties that define what step they are APPLICABLE_FOR, if they are ENABLED to be used, if they can be automatically applied (AUTO), or what step the suggestions was GENERATED_AT. A suggestion must be ENABLED and the APPLICABLE_FOR synthesis or implementation step run in order for a suggestion to be applied to the design.

You can write the suggestions from the design into an RQS file using the `write_qor_suggestions` command. After resetting the design flow to the appropriate step, you can read suggestions back into the design using the `read_qor_suggestions` command, and then run the synthesis or implementation step to apply the enabled suggestions.

The recommended method for working with QoR suggestions is:

1. Run the report to create recommendations (`report_qor_suggestions`)

2. Write the suggestions to an RQS file on disk (`write_qor_suggestions`).

3. Reset the design to the appropriate step.

4. Read the RQS file into the design to restore the suggestions (`read_qor_suggestions`).

5. Run the synthesis or implementation step to apply the suggestion (`synth_design`, `opt_design`...).

For strategy suggestions, a machine learning based analysis is conducted and the best 3 strategies are reported. This analysis is only conducted on a routed design that has been implemented with

- the `Default` or `Explore` directive for `opt_design` and

- `place_design`, `phys_opt_design`, and `route_design` must have the directive set to `Default` or `Explore`. Mix and match of directives is not allowed within these commands.

Strategy suggestions do not apply to synthesis. Unlike the non-strategy suggestions, only one strategy suggestion can be applied to each run.

The recommended method for working with QoR strategy suggestions is:

1. Run the report to create strategies and other recommendations(`report_qor_suggestions`).

2. Write the suggestions to a directory disk. There is one RQS file per strategy (`write_qor_suggestions -strategy_dir`).

3. Update all the directive settings to -directive RQS for `opt_design`, `place_design`, `phys_opt_design`, and `route_design` commands.

4. For each run, read one RQS file into the design. This will contain the strategy suggestion and all exported non-strategy suggestions(`read_qor_suggestions`).

5. Run the implementation step from the `opt_design` command to apply the strategy and all the design suggestions.

Arguments

`-file <arg>` - (Optional) Write the QoR report into the specified file. The specified file will be overwritten if one already exists, unless `-append` is also specified.

`-csv <arg>` - (Optional) Write a CSV file that shows the timing path and analysis features. The specified file will be overwritten if one already exists.

Note: If the path is not specified as part of the file name for either `-file` or `-csv`, the file will be written into the current working directory that can be identified using the `[pwd]` Tcl command.

`-name <arg>` - (Optional) The name of the view to display in the Vivado IDE when run in GUI mode. Only one QoR suggestion report can be opened in the Vivado tool at one time. This makes it different from most other available reports. If there is already an open report in the Report view, that report will be closed and a new report is opened.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-return_string` - (Optional) Direct the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-max_strategies <args>` - (Optional) Specify the maximum number of ML strategies returned. The algorithm may generate less than the maximum specified number if not enough suitable strategies are identified. Default is 3.

`-max_paths <arg>` - (Optional) Specify the number of critical paths to analyze. The default is the 100 worst timing paths per timing group. Timing paths that have a WNS > 0.000 are ignored unless -report_all_suggestions is set.

`-no_split` - (Optional) Do not split the lines in the table when generating the report.

`-report_all_suggestions` - (Optional) Relaxes the criteria that suppresses suggestions.

- Utilization suggestions do not need to save a critical resource

- Congestion suggestions are offered on designs when timing is closed

- Timing suggestions on paths that have a WNS < +0.500

`-cells <arg>` - (Optional) Report QoR suggestions for the specified hierarchical cells and all data paths passing through them.

`-of_objects` <args> - (Optional) Report existing suggestions defined by the specified list of QoR objects as returned by the `get_qor_suggestions` command. When `-of_objects` is specified, the report does not analyze the design to identify new suggestions, it merely reports the specified existing suggestion objects.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

Report suggestions after analyzing the worst 10 paths.

```
report_qor_suggestions -max_paths 10
```

Report existing suggestions in the design from a prior run of `report_qor_suggestions`.

```
report_qor_suggestions -of_objects [get_qor_suggestions]
```

See Also

- get_qor_suggestions
- read_qor_suggestions
- report_qor_assessment
- write_qor_suggestions

Send Feedback

report_ram_utilization

Report configuration about RAMs in design

Syntax

```
report_ram_utilization [-append] [-file <arg>] [-csv <arg>]
    [-return_string] [-cells <args>] [-include_path_info] [-include_lutram]
    [-quiet] [-verbose]
```

Returns

Report

Usage

Name	Description
[-append]	Append to existing file
[-file]	Filename to output results to. (send output to console if -file is not used)
[-csv]	Write detailed tables to the CSV filename specified
[-return_string]	return report as string
[-cells]	Limits the reporting to only those memory arrays that are contained within the specified cells.
[-include_path_info]	Adds path info to the RAM inputs/outputs
[-include_lutram]	Adds LUTRAMs to the report
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Report

Description

Reports RAM usage in the current design at any stage after synthesis. It is recommended to run this report after optimizations in opt_design have been run. The report is returned to the standard output, unless the -file or -return_string arguments are used.

The report details the following tables:

- **Summary**: The Summary table shows the utilization by RAM type, broken down into URAM, Block RAM and LUTRAM. For each RAM type, it is broken down into the individual primitives. Also detailed is the inferred primitive %. If using a synthesis tool other than Vivado Synthesis or instantiation of primitives, then rams will be reported under instantiated primitives.

The information in the summary can be used to determine if there are imbalances in the utilization of different types of RAMs. Based on the summary, a design might benefit from retargeting heavily used primitives to lesser used ones at the cost of being less efficient.

Additionally LUTRAM implementation can be evaluated. LUTRAM primitives vary in their efficiency of Bits/LUT. For example, a RAM32M16 uses 8 LUTs and can have up to 14 data bits. Using RAM32X1D primitives, 8 LUTs yields a maximum of 8 bits when LUT combined.

- **Memory Description**: The Memory Description table gives an RTL look at the inferred memories. It refers to memories that are inferred directly or indirectly using XPMs. The table is ordered by the total number of bits in the memory array. Items reported are memory type and mapping.

  The mapping format is `<report>:<physical>:<depth>x<width>`

The following tables are repeated for inferred and instantiated primitives.Collectively they are refered to as the detailed tables.

- **Memory Utilization**: The Memory Utilization table details how each memory array is mapped to primitives. Also provided is information on how efficient the mapping is and whether the mapping is limited by depth or width.

- **Memory Performance**: The Memory Performance table details performance aspects of the RAM. It includes information whether the output register is used, cascading has been used. If the `-include_path_info` switch is specified there is extra information on the worst path from each pin listed.

- **Memory Power**: The Memory Power table details what power aspects of the RAM have been used. It includes information on cascading and whether the enable pin is tied to POWER or a signal.

This command returns the requested report to the Tcl console, to a file, or as a string; or returns an error if it fails.

Arguments

`-file <arg>` - (Optional) Write the report into the specified file. Send output to console if not specified. This command will overwrite any files of the same name without warning.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-csv <arg>` - (Optional) Write the report into the specified CSV file.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option is only valid for the text report specified using the `-file` option and not the CSV file.

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-cells <arg>` - (Optional) Report the RAM resources utilized by one or more hierarchical cells in the current design. The cells must be specified as objects, using `get_cells`, rather than by name.

`-include_lutram` - (Optional) Include LUTRAMs in the detailed tables. By default, only BlockRAMs and UltraRAMs are reported.

`-include_path_info` - (Optional) Include a shorthand representation of the worst case timing path from the given pins under the Memory Performance table of the report.

Note: This option can add significant runtime to the report.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example provides a full report of the inferred and non-inferred BlockRAM and UltraRAM resources and writes the results to both the specified text and CSV files:

```
report_ram_utilization -csv ./ram_util.csv -file ./ram_util.txt
```

The following example provides a full report of the inferred and non-inferred BlockRAM, UltraRAM and LUTRAM resources and writes the results to the specified CSV file:

```
report_ram_utilization -csv ./ram_util.csv -include_lutram
```

See Also

- get_cells
- report_utilization

report_route_status

Report on status of the routing.

Syntax

```
report_route_status [-return_nets] [-return_string] [-file <arg>] [-append]
    [-of_objects <args>] [-route_type <arg>] [-list_all_nets] [-show_all]
    [-dump_routes] [-has_routing] [-boolean_check <arg>] [-ignore_cache]
    [-quiet] [-verbose]
```

Usage

Name	Description
[-return_nets]	Return a list of nets that match the given -route_type type
[-return_string]	Set the result of running the report in the Tcl interpreter's result variable
[-file]	Filename to output results to. (send output to console if -file is not used)
[-append]	Append the results to file, don't overwrite the results file
[-of_objects]	Report detailed routing for these routes
[-route_type]	Only show routes with the given status: UNPLACED|UNPLACED_ALL|UNROUTED|PARTIAL|GAPS|CONFLICTS|ANT ENNAS|NODRIVER|MULTI_DRIVER|LOCKED_NODES|HAS_ROUTING|INTR ASITE|NOLOADS|HIERPORT|ROUTED (ignored if -of_objects is used)
[-list_all_nets]	list full route information for every net in the design (ignored if -of_objects is used)
[-show_all]	list all relevant pins for routes marked as UNPLACED or PARTIAL routes and list all relevant nodes for routes marked as ANTENNAS or CONFLICTS routes (by default only the first 15 pins or nodes are listed for a route)
[-dump_routes]	show the full routing tree for every routed net in the design. This is VERY VERBOSE.
[-has_routing]	returns 0 if there is no routing currently stored for this design and 1 if there is. All other options are ignored.
[-boolean_check]	returns 1 if the given flag is true and 0 if it is not. Value flags that can be checked are: PLACED_FULLY|PARTIALLY_ROUTED|ROUTED_FULLY|ERRORS_IN_ROUT ES. All other options are ignored (cannot be used with -has_routing).
[-ignore_cache]	throw away all cached information and recalculate the route status for the entire design (slow)
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Report

Description

Reports the state of routing in the current design.

The route status report can include a wide range of information, from a simple 1 if the design has routing, to a complete route tree for each net in the design.

Arguments

`-return_nets` - (Optional) Returns a list of nets that can be processed by other Tcl commands. When the -return_nets argument is used, it is required to also use the -route_type argument with a valid route state.

Note: This argument cannot be used with `-return_string, -of_objects, -list_all_nets, -dump_routes, -has_routing,` or `-boolean_check arguments`. This argument will not generate the `report_route_status` output.

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-file <arg>` - (Optional) Write the report into the specified file. The specified file will be overwritten if one already exists, unless `-append` is also specified.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-of_objects <args>` - (Optional) Report the full routing tree for the specified route, net, or xdef_net objects.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-route_type <arg>` - (Optional) Only show routes with the specified route status. Valid route states are:

- ANTENNAS - the net has antennas or islands. Antennas are connected routing that don't go to a valid load, islands are pieces of disconnected routing in the route tree.

- CONFLICTS - the net has conflicts (either multiple routes share the same node, or a site internally has a logical net driving a site pin and that site pin has routing to it from a different logical net, or a routed site reported that it was invalid).

Send Feedback

- HIERPORT - the net has either no loads or no drivers and is also connected to a top level terminal port.

- INTRASITE (or INTERNAL) - is fully routed inside a site. No routing required.

- GAPS - the net has known gaps in the route tree.

- HAS_ROUTING - the net has a route tree associated with it.

- LOCKED_NODES (or FIXED) - the net has locked routing nodes in it.

- NODRIVER (or NO_DRIVERS) - the net has no drivers.

- NOLOADS (or NO_LOADS) - the net has no loads and so needs no routing.

- PARTIAL - the net has routing but is not fully routed.

- ROUTED - the net is fully routed with no errors.

- UNPLACED (or UNPLACED_PINS) - the net has any unplaced pins.

- UNPLACED_ALL - none of the pins for the net are placed.

- UNROUTED - the net needs to be routed but has no route data.

Note: The `-route_type` option is ignored if `-of_objects` is also specified.

`-list_all_nets` - (Optional) Report summary route status for every net in the design, or those selected by `-route_type` if it is given.

Note: This option is ignored if `-of_objects` is specified.

`-show_all` - (Optional) Report all relevant pins for routes marked as UNPLACED or PARTIAL routes and list all relevant nodes for routes marked as ANTENNAS or CONFLICTS routes. As a default only the first 15 pins or nodes are listed for a given route.

`-dump_routes` - (Optional) Report the full routing tree for every routed net in the design.

Note: This is a very long report, and can take some time to generate.

`-has_routing` - (Optional) Returns false (0) if the design is unrouted, and returns true (1) if the design has routing. All other options are ignored when `-has_routing` is specified.

Note: Has routing does not mean fully routed.

`-boolean_check <arg>` - (Optional) Returns 1 if the specified design attribute is true, and returns 0 if it is not true. Design attributes that can be checked include: PLACED_FULLY, PARTIALLY_ROUTED, ROUTED_FULLY, ERRORS_IN_ROUTES.

Note: This option cannot be used with `-has_routing` or `-return_nets`.

`-ignore_cache` - (Optional) By default the `report_route_status` command is iterative, and only updates the route information for new nets and routes as the design is implemented. This argument will cause the command to ignore the cached information and regenerate the report for the entire design.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example reports the route status for the specified nets:

```
report_route_status -of_objects [get_nets u4*]
```

The following example returns a list of nets that have routing but are not fully routed to the Find Results window in the Vivado IDE.

```
show_objects [report_route_status -return_nets -route_type PARTIAL]
```

See Also

- route_design

report_scopes

Print names of the children scopes (declarative regions) of given scope(s) or the current scope

Syntax

```
report_scopes [-quiet] [-verbose] [<hdl_scopes>...]
```

Returns

Report_scopes prints a subset of properties of the HDL scope on console in textual format.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<hdl_scopes>]	The hdl_objects to report. Default is report_scopes [get_scopes *]

Categories

Simulation

Description

Reports the names and types of HDL Scopes in the current scope of the current simulation, or of specified scopes.

An HDL Scope is a declarative region of an HDL file, where objects are declared. The following are examples of HDL Scopes in Verilog and VHDL:

- Verilog scopes: module, function, task, process, other begin-end blocks

- VHDL scopes: entity/architecture pair, block, function, procedure, process

You must have an open simulation to use this command.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the set_msg_config command.

<hdl_scopes> - (Optional) Specifies the scopes upon which to report. The default is the current scope.

Examples

The following example reports the children scopes of /tb/UUT:

```
report_scopes [get_scopes /tb/UUT/* filter {type==module}
```

The following example reports the children scopes of the current scope:

```
report_scopes
         VHDL Instance: {U_DEBOUNCE_0}
         VHDL Instance: {U_DEBOUNCE_1}
         VHDL Instance: {U_SINEGEN}
         VHDL Instance: {U_FSM}
          VHDL Process: {line__138}
          VHDL Process: {line__184}
          VHDL Process: {line__185}
          VHDL Process: {line__186}
          VHDL Process: {line__187}
          VHDL Process: {line__191}
```

See Also

- current_scope
- get_scopes

report_sim_device

Report the list of correct SIM_DEVICE attribute values for cell types in the target part

Syntax

```
report_sim_device [-part <arg>] [-file <arg>] [-append] [-return_string]
    [-quiet] [-verbose]
```

Returns

Report

Usage

Name	Description
[-part]	Part
[-file]	Output file
[-append]	Append the results to file
[-return_string]	Return report as string
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Report

Description

Cells with the SIM_DEVICE attribute may have slightly different behavior when applied in different families of devices. The value of SIM_DEVICE directs the simulator to match the behavior of the cell to the target device. Use report_sim_device to report the correct SIM_DEVICE attribute value to use for cell types in the target part.

> **IMPORTANT!** *Unless `-part` is specified, the currently loaded part is used as the target part for this command.*

Arguments

`-part <arg>` - (Optional) Target part

`-file <arg>` - (Optional) Write the SIM_DEVICE report into the specified file. The specified file will be overwritten if one already exists, unless -append is also specified.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-return_string` - (Optional) Directs the output to a Tcl string. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the -file option.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example generates the report of `SIM_DEVICE` attributes and correct values to use for the `xcvu065-ffvc1517-1-i` part.

```
report_sim_device -part xcvu065-ffvc1517-1-i
```

See also

- report_sim_version
- get_primitives
- get_parts

Send Feedback

report_sim_version

Report the list of correct SIM_VERSION attribute values for cell types in the target part

Syntax

```
report_sim_version [-part <arg>] [-file <arg>] [-append] [-return_string]
    [-unisim <arg>] [-quiet] [-verbose]
```

Returns

Report

Usage

Name	Description
[-part]	Part
[-file]	Output file
[-append]	Append the results to file
[-return_string]	Return report as string
[-unisim]	Return the SIM_VERSION for the unisim if present
[-quiet]	quiet option
[-verbose]	Suspend message limits during command execution

Categories

[Report](#)

report_simlib_info

Report info of simulation libraries

Syntax

```
report_simlib_info [-file <arg>] [-append] [-quiet] [-verbose] <path>
```

Usage

Name	Description
[-file]	Output file Default: report_simlib_info.log
[-append]	Append mode
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<path>	Specify the path for pre-compiled libraries

Categories

Simulation

Description

Report information on libraries compiled by the `compile_simlib` command.

Arguments

`-file <arg>` - (Optional) Write the report into the specified file. The specified file will be overwritten if one already exists, unless `-append` is also specified.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

Send Feedback

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<path> - (Required) The path to the compiled simulation library.

Examples

The following example reports information related to the compiled simulation library at the specified path:

```
report_simlib_info C:/Data/compiled_simlib
```

See Also

- compile_simlib

report_ssn

Run SSN analysis on the current package and pinout

Syntax

```
report_ssn [-name <arg>] [-return_string] [-format <arg>] [-file <arg>]
    [-append] [-phase] [-quiet] [-verbose]
```

Returns

Ssn report

Usage

Name	Description
[-name]	Output the results to GUI panel with this name
[-return_string]	Return report as string
[-format]	Report format. Valid arguments are CSV, HTML, TXT Default: csv
[-file]	Filename to output results to. (send output to console if -file is not used)
[-append]	Append the report to the specified file
[-phase]	Account for multi-clock phase in the analysis
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Report

Description

Perform a simultaneous switching noise (SSN) analysis of the current design. The SSN analysis is an accurate method for predicting how output switching affects interface noise margins. The calculation and estimates are based on a range of variables intended to identify potential noise-related issues in your design and should not be used as final design "sign off" criteria.

SSN analysis provides estimates of the disruption that simultaneously switching outputs can cause on other output ports in the I/O bank. The SSN predictor incorporates I/O bank-specific electrical characteristics into the prediction to better model package effects on SSN.

Send Feedback

The `report_ssn` command can be affected by the temperature grade of the selected device as defined by the `-grade` option of the `set_operating_condition` command. Setting the temperature grade prior to running noise analysis lets you see how noisy signals can be on Commercial, Extended, Industrial, Q-Grade, or Military grade devices.

By default, `report_ssn` assumes that every port toggles asynchronously. This results in a worst-case noise analysis, which may be overly pessimistic. The `-phase` option lets you consider clocking information available in the design to more accurately report SSN noise. Clocks must be defined using the `create_clock` and `create_generated_clock` commands. The period, phase shift and duty cycle of the generated clocks have significant impact on SSN analysis.

The `report_ssn` command provides a detailed SSN analysis for Xilinx UltraScale architecture devices, Virtex-7, Kintex-7, and Artix-7 devices. The report is returned to the standard output, unless the `-file`, `-return_string`, or `-name` arguments are specified.

> **TIP:** *Not all parts support the `report_ssn` command. The Vivado Design Suite will return an error if you run `report_ssn` on a target part that does not support SSN analysis. You can query the SSN_REPORT property of a part to see if it supports the command. Refer to the Examples for more information.*

Arguments

`-name <arg>` - (Optional) Specifies the name of the results to output to the GUI.

`-return_string` - (Optional) Directs the output to a Tcl string. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-format [ CSV | HTML | TXT ]` - (Optional) Specifies the format of the output as either comma-separated values (CSV), HTML, or an ASCII (TXT) file. The default output is CSV.

Note: The format applies when `-file` is specified, but is otherwise ignored.

`-file <arg>` - (Optional) Write the SSN report into the specified file. The specified file will be overwritten if one already exists, unless `-append` is also specified.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-append` - Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-phase` - (Optional) Consider clock switching cycles in SSN analysis to provide a more accurate result.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example performs an SSN analysis on the current design, formats the output as HTML, and writes the output to the specified file:

```
report_ssn -format html -file C:/Data/devSSN.html
```

The following example performs an SSN analysis, with phase analysis, and returns the output to a string which is stored in the specified variable:

```
set devSSN [report_ssn -phase -format html -return_string]
```

Note: The `-format` argument in the preceding example is ignored in the absence of `-file`.

The following example queries the part in the current project to see if it supports the `report_ssn` command, and then gets a list of parts from the same part family that support the command:

```
get_property SSN_REPORT [get_property PART [current_project]]
get_parts -filter "FAMILY == [get_property FAMILY [get_property PART \
[current_project]]] && SSN_REPORT"
```

See Also

- create_clock
- create_generated_clock
- get_parts
- get_property
- reset_ssn
- report_property
- set_operating_conditions

report_stacks

Print names of processes in a design, which are waiting inside a subprogram, in textual format

Syntax

```
report_stacks [-of_instance <arg>] [-quiet] [-verbose]
```

Returns

Returns string

Usage

Name	Description
[-of_instance]	Default: NULL
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Simulation

Description

Print name of the HDL process scopes waiting inside a subprogram in textual format.

Arguments

`-of_instance` - (Optional) Specifies the instance to report the stacks from.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

Example design:

```
module top;
     int i;
     function void f(input int in1);
          automatic int a;
          a = in1 + 7;
          $display($time, " in f :: a %d in1 %d ", a, in1);
     endfunction
     task automatic t(input int in2);
          int b;
          b = in2 + 10;
          $display($time, " in t :: in2 %d b %d ", in2, b);
          #5;
          f(b);     // Case C
          $display($time, " Back in t : after wait and f(%d) ", b);
     endtask
     initial begin                        // "/top/Initial18_0"
          $display($time, " in initial 1 ");
          i = 200;
          t(i);          // Case B
          $display($time, " Back in initial 1 after t(%d) ", i);
     end
     initial begin                        // "/top/Initial25_1"
          $display($time, " in initial 2 ");
          #2;
          f(50);         // Case A
          $display($time, " Back in initial 2 after f(50) ");
     end
  endmodule
```

When simulation is stopped inside function "f" for its call at 'Case A' , the two processes `/top/Initial18_0` and `/top/Initial25_1`, are waiting inside task "t" (call at 'CaseB') and function "f" (call at 'CaseA') respectively.

```
1. > report_stacks
   Verilog Process: {/top/Initial18_0}
   Verilog Process: {/top/Initial25_1}
```

See Also

- get_stacks

report_switching_activity

Get switching activity on specified objects

Syntax

```
report_switching_activity [-static_probability] [-signal_rate]
    [-toggle_rate] [-default_static_probability] [-default_toggle_rate]
    [-average] [-file <arg>] [-return_string] [-append] [-hier] [-all]
    [-type <args>] [-quiet] [-verbose] [<objects>...]
```

Usage

Name	Description
[-static_probability]	report static probability
[-signal_rate]	report signal rate
[-toggle_rate]	report toggle rate
[-default_static_probability]	report default static probability
[-default_toggle_rate]	report default toggle rate
[-average]	Report the average switching activity for the specified objects.
[-file]	Filename to output results to. (send output to console if -file is not used)
[-return_string]	return switching activity as string
[-append]	append switching activity to end of file
[-hier]	Hierarchically reports the switching activity on nets within a hierarchical instance provided via <objects> option.
[-all]	Report switching activities for all nets for the design.
[-type]	Specify nodes in a specific category. List of valid type values: io_output, io_bidir_enable, register, lut_ram, lut, dsp, bram_enable, bram_wr_enable, gt_txdata, gt_rxdata.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<objects>]	objects

Categories

Report

Description

This command is used to report different kinds of switching activity on design nets, ports, pins, and cells in the current synthesized or implemented design. These include simple signal rate and simple static probability on nets, ports, and pins; and state dependent static probabilities on cells.

The reported values are defined using the `set_switching_activity` command.

Send Feedback

Note: This command returns the switching activity for the specified objects, or the current design.

By default the report is written to the Tcl console or STD output. However, the results can also be written to a file or returned as a string if desired.

Arguments

`-static_probability` - (Optional) Specifies that the command returns static probability as part of the report.

`-signal_rate` - (Optional) Specifies that the command returns the signal rate as part of the report.

`-toggle_rate` - (Optional) Reports the toggle rate (%) as the switching rate of the output of synchronous logic elements compared to a given clock input.

`-average` - (Optional) Reports the average switching activity for the specified objects.

`-default_static_probability` - (Optional) Reports the default static probability to be used in power analysis on the current design. The default static probability is set using the `set_switching_activity` command.

Note: This option does not require objects to be specified since the default applies to the current design.

`-default_toggle_rate` - (Optional) Reports the default toggle rate to be used in power analysis on the primary inputs of the current design. You can define the default toggle rate using the `set_switching_activity` command.

Note: This option does not require objects to be specified since the default applies to the current design.

`-file` <filename> - (Optional) Writes the report to the specified path and file.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-append` - (Optional) Appends the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-return_string` - (Optional) Returns the data as a text string for assignment to a Tcl variable.

`-hier` - (Optional) Reports the switching activity hierarchically for signals in the specified hierarchical <objects>. Without `-hier`, the switching activity is applied to the specified <objects> at the current level of the hierarchy.

`-all` - (Optional) Must be used with `-type`, report the switching activity on nets within all instances specified by `-type`.

Send Feedback

`-type <arg>` - (Optional) Reports the switching activity for the specified type of logic entity. By default, the command is applied to the top-level of the current design, or to the specified `<objects>`. The `-type` option applies the command settings to the specified type of logic objects in the top-level of the current design. The `-all` option or `-hier` option can be used to modify the scope of objects the command applies to. Valid logic types include:

- io_output - Primary outputs.

- io_bidir_enable - Enable pin of Bidir ports.

- register - All register outputs in the design/hierarchy specified.

- lut - All LUT outputs in the design/hierarchy specified.

- lut_ram - All distributed ram outputs in the design/hierarchy specified.

- dsp - All DSP outputs in the design/hierarchy specified.

- bram_enable - Enable pins (ENARDEN/ENBWREN) of BRAMs.

- bram_wr_enable - Write enables of BRAMs (WEA/WEBWE).

- gt_txdata - Output TX data pins of all GTs.

- gt_rxdata - Output RX data pins of all GTs.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<objects>` - (Optional) A list of port, pin, and net objects to report the switching activity of; or a list of cells when specified with `-type` to define logic objects.

Examples

The following example reports the signal_rate and static probability value on all output ports:

```
report_switching_activity -signal_rate -static_probability [all_outputs]
```

The following example reports the signal_rate and static probability value on all LUT objects in the design:

```
report_switching_activity -signal_rate -static_probability -type lut -all
```

The following example reports the average toggle rate on all output ports:

```
report_switching_activity -toggle_rate -average [all_outputs]
```

See Also

- power_opt_design
- report_power
- reset_switching_activity
- set_switching_activity

report_synchronizer_mtbf

Compute mean time between failures and display report

Syntax

```
report_synchronizer_mtbf [-file <arg>] [-append] [-return_string]
    [-warn_if_mtbf_below <arg>] [-quiet] [-no_header] [-report_endpoints]
    [-verbose]
```

Returns

Report

Usage

Name	Description
[-file]	Filename to output results to. (send output to console if -file is not used)
[-append]	Append the results to file, don't overwrite the results file
[-return_string]	Return the report output as a string
[-warn_if_mtbf_below]	Default: 1e+12
[-quiet]	Ignore command errors
[-no_header]	Report without the header
[-report_endpoints]	Report cdc path end points
[-verbose]	Suspend message limits during command execution

Categories

Report, Timing

Description

Note: This command is supported for Xilinx UltraScale devices only, and does not support 7 series devices.

The `report_synchronizer_mtbf` command reports mean time between failures (MTBF) of each clock domain crossing (CDC) synchronizer chain in a design, and provides an overall MTBF covering all synchronizers. Synchronizer registers must have ASYNC_REG properties with value TRUE to be properly identified as synchronizers for reporting.

Asynchronous clock domain crossings (CDCs) can fail due to metastability as data is captured asynchronously and may settle to different values on different loads in the circuit. Synchronizer registers are used to improve overall circuit reliability for designs which contain multiple clock domains, in which asynchronous data transfers occur, or in which external asynchronous signals are captured with an internal clock. A synchronizer chain consists of two or more registers

Send Feedback

connected sequentially with the first stage capturing the data signal from the asynchronous clock domain. The successive register stages provide additional settling time for metastable events and increase MTBF. The synchronizer registers must have ASYNC_REG properties with values of TRUE. Besides reporting MTBF, the ASYNC_REG properties instruct synthesis, simulation and implementation tools to optimize for increased MTBF and improve overall behavior of the synchronizer circuit.

> **TIP:** *Avoid using different set/reset or clock enable control signals on registers within a synchronizer chain.*

This command returns the MTBF report, or returns an error if it fails. The command issues a warning message when the MTBF cannot be calculated correctly, for example when a CDC is improperly constrained. The following conditions result in an UNDEFINED synchronizer MTBF value:

- One or both clocks of the CDC are unconstrained.

- There is a timing violation involving registers in the synchronizer chain.

- There is a zero toggle rate detected for the CDC data.

In the case of a zero toggle rate, it may be necessary to use the `set_switching_activity` command to manually override the toggle rate on the CDC net with a realistic value. This involves assigning the Toggle Rate and the Static Probability:

- Toggle Rate: The number of CDC data signal transitions measured in Million Transitions per Second.

- Static Probability: The percentage of time during which the CDC data signal is driven at a high logic level.

Example: to assign a toggle rate of 12.5% with 0.5 static probability on a CDC net named resync[0]:

```
set_switching_activity -toggle_rate 12.5 -static_probability 0.5 \
[get_nets resync[0]]
```

The report contents include the following data for each synchronizer chain in the design:

- MTBF: The Mean Time Between Failures for the CDC synchronizer reported in dynamic time units, from seconds to years. An invalid MTBF value is reported as UNDEFINED.

- Data Toggle Rate: The rate at which the CDC data switches, based on the default switching activity for the design as reported by `report_switching_activity`. Measured in (Mts) Millions of Transitions per Second. The rate can be overridden using the `set_switching_activity` command targeting the CDC net object.

- Data Sample Rate: The rate at which the CDC data is sampled, equivalent to the synchronizer chain frequency, measured in MHz.

- Settling Time: The total amount of positive slack in nanoseconds on the timing paths from synchronizer register outputs. Higher Settling Time increases MTBF.

- Sending Domain: The clock domain of the source of the CDC data. A value of UNCONSTRAINED is reported if the source clock is not defined.

- Receiving Domain: The clock domain of the destination of the CDC data. A value of UNCONSTRAINED is reported if the destination clock is not defined.

- Number of Stages: This is the length of the synchronizer chain, which equals the number of registers with ASYNC_REG value of TRUE. The MTBF calculation will determine the likelihood that the output register or registers (should the fanout be greater than 1) will experience a metastable event. For example in a typical synchronizer containing 2 registers with the ASYNC_REG property set, the MTBF calculation indicates the probability that the output register(s) following the last ASYNC_REG register will capture an incorrect value resulting from the metastable event. When a synchronizer is connected to more than 1 output register, the minimum slack from all the paths will be used in the MTBF calculation to ensure that all registers capture the same logic level.

- CDC Net Name: This is the logical net name of the CDC data, the data that is captured asynchronously.

This command returns the MTBF report, or returns an error if it fails.

The report also includes an overall MTBF calculated using the MTBF of all synchronizers in the design, calculated as the inverse of the sum of the reciprocals of the individual synchronizer MTBF values: $(1 / (1/MTBF\_1 + 1/MTBF\_2 + ... + 1/MTBF\_N))$ for N synchronizers.

Arguments

`-file <arg>` - (Optional) Write the report into the specified file. This command will overwrite any files of the same name without warning.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-warn_if_mtbf_below <arg>` - (Optional) Specify a value as a floating point number, below which the Vivado Design Suite will issue a warning in addition to the report. The default value is 1e+12.

`-no_header` - (Optional) Write the report without the addition of the standard header.

`-report_endpoints` - (Optional) Report the total number of CDC path endpoints. This is the sum of Safe, Unsafe, and Unknown endpoints.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example writes the MTBF report to the specified file:

```
report_synchronizer_mtbf -file C:/Data/mtbf_report.txt
```

See Also

- get_nets
- report_cdc
- report_clock_interaction
- report_clock_networks
- report_switching_activity
- set_switching_activity

Send Feedback

report_timing

Report timing paths

Syntax

```
report_timing [-from <args>] [-rise_from <args>] [-fall_from <args>]
    [-to <args>] [-rise_to <args>] [-fall_to <args>] [-through <args>]
    [-rise_through <args>] [-fall_through <args>] [-delay_type <arg>]
    [-setup] [-hold] [-max_paths <arg>] [-nworst <arg>] [-unique_pins]
    [-path_type <arg>] [-input_pins] [-no_header] [-no_reused_label]
    [-slack_lesser_than <arg>] [-slack_greater_than <arg>] [-group <args>]
    [-sort_by <arg>] [-no_report_unconstrained] [-user_ignored]
    [-of_objects <args>] [-significant_digits <arg>] [-column_style <arg>]
    [-file <arg>] [-append] [-name <arg>] [-no_pr_attribute]
    [-routable_nets] [-return_string] [-warn_on_violation] [-cells <args>]
    [-rpx <arg>] [-quiet] [-verbose]
```

Usage

Name	Description
[-from]	From pins, ports, cells or clocks
[-rise_from]	Rising from pins, ports, cells or clocks
[-fall_from]	Falling from pins, ports, cells or clocks
[-to]	To pins, ports, cells or clocks
[-rise_to]	Rising to pins, ports, cells or clocks
[-fall_to]	Falling to pins, ports, cells or clocks
[-through]	Through pins, ports, cells or nets
[-rise_through]	Rising through pins, ports, cells or nets
[-fall_through]	Falling through pins, ports, cells or nets
[-delay_type]	Type of path delay: Values: max, min, min_max, max_rise, max_fall, min_rise, min_fall Default: max
[-setup]	Report max delay timing paths (equivalent to -delay_type max)
[-hold]	Report min delay timing paths (equivalent to -delay_type min)
[-max_paths]	Maximum number of paths to output when sorted by slack, or per path group when sorted by group: Value >=1 Default: 1
[-nworst]	List up to N worst paths to endpoint: Value >=1 Default: 1
[-unique_pins]	for each unique set of pins, show at most 1 path per path group
[-path_type]	Format for path report: Values: end, summary, short, full, full_clock, full_clock_expanded Default: full_clock_expanded
[-input_pins]	Show input pins in path
[-no_header]	Do not generate a report header
[-no_reused_label]	Do not label reuse status on pins in the report
[-slack_lesser_than]	Display paths with slack less than this Default: 1e+30

Send Feedback

Name	Description
[-slack_greater_than]	Display paths with slack greater than this Default: -1e+30
[-group]	Limit report to paths in this group(s)
[-sort_by]	Sorting order of paths: Values: group, slack Default: slack
[-no_report_unconstrained]	Do not report infinite slack paths
[-user_ignored]	Only report paths which have infinite slack because of set_false_path or set_clock_groups timing constraints
[-of_objects]	Report timing for these paths
[-significant_digits]	Number of digits to display: Range: 0 to 3 Default: 3
[-column_style]	style for path report columns: Values: variable_width, anchor_left, fixed_width Default: anchor_left
[-file]	Filename to output results to. (send output to console if -file is not used)
[-append]	Append the results to file, don't overwrite the results file
[-name]	Output the results to GUI panel with this name
[-no_pr_attribute]	for Dynamic Function eXchange designs, do not report whether netlist resources are in the static or reconfigurable regions
[-routable_nets]	store the number of routable nets traversed as a property on timing paths.
[-return_string]	return report as string
[-warn_on_violation]	issue a critical warning when the report contains a timing violation
[-cells]	run report_timing on the specified cell(s)
[-rpx]	Filename to output interactive results to.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Report, Timing

Description

⭐ **IMPORTANT!** *If the design has no timing constraints, report_timing reports on unconstrained paths in the design. However, if even one path has timing constraints then report_timing only reports on the constrained paths in the design, unless unconstrained timing paths are specified by the -from/-to options.*

This command performs timing analysis on the specified timing paths of the current Synthesized or Implemented Design. By default the tool reports the timing path with the worst calculated slack within each path group. However, you can optionally increase the number of paths and delays reported with the use of the -nworst or -max_paths arguments.

💡 **TIP:** *The report_timing can be multi-threaded to speed the process. Refer to the set_param command for more information on setting the general.maxThreads parameter.*

The timing engine runs in "quad" timing mode, analyzing min and max delays for both slow and fast corners. You can configure the type of analysis performed by the `config_timing_corners` command. However, it is not recommended to change the default because this reduces the timing analysis coverage.

Note: By default the report is written to the Tcl console or STD output. However, the results can also be written to the GUI, to a file, or returned as a string if desired.

Arguments

`-from` <args> - (Optional) The starting points of the timing paths to be analyzed. Ports, pins, or cells can be specified as timing path startpoints. You can also specify a clock object, and all startpoints clocked by the named clock will be analyzed.

`-rise_from` <args> - (Optional) Similar to the `-from` option, but only the rising edge of signals coming from the startpoints are considered for timing analysis. If a clock object is specified, only the paths launched by the rising edge of the clock are considered as startpoints.

`-fall_from` <args> - (Optional) Similar to the `-from` option, but only the falling edge of signals coming from the startpoints are considered for timing analysis. If a clock object is specified, only the paths launched by the falling edge of the clock are considered as startpoints.

`-to` <args> - (Optional) The endpoints, or destination objects of timing paths to be analyzed. Ports, pins, and cell objects can be specified as endpoints. A clock object can also be specified, in which case endpoints clocked by the named clock are analyzed.

`-rise_to` <args> - (Optional) Similar to the `-to` option, but only the rising edge of signals going to the endpoints is considered for timing analysis. If a clock object is specified, only the paths captured by the rising edge of the named clock are considered as endpoints.

`-fall_to` <args> - (Optional) Similar to the `-to` option, but only the falling edge of signals going to the endpoints is considered for timing analysis. If a clock object is specified, only the paths captured by the falling edge of the named clock are considered as endpoints.

`-through` <args> - (Optional) Consider only paths through the specified pins, cell instance, or nets during timing analysis. You can specify individual `-through` (or `-rise_through` and `-fall_through`) points in sequence to define a specific path through the design for analysis. The order of the specified through points is important to define a specific path. You can also specify through points with multiple objects, in which case the timing path can pass through any of the specified through objects.

`-rise_through` <args> - (Optional) Similar to the `-through` option, but timing analysis is only performed on paths with a rising transition at the specified objects.

`-fall_through` <args> - (Optional) Similar to the `-through` option, but timing analysis is only performed on paths with a falling transition at the specified objects.

`-delay_type <arg>` - (Optional) Specifies the type of delay to analyze when running the timing report. The valid values are min, max, min_max, max_rise, max_fall, min_rise, min_fall. The default setting for `-delay_type` is max.

`-setup` - (Optional) Check for setup violations. This is the same as specifying `-delay_type max`.

`-hold` - (Optional) Check for hold violations. This is the same as specifying `-delay_type min`.

> **TIP:** `-setup` *and* `-hold` *can be specified together, which is the same as specifying* `-delay_type min_max`.

`-max_paths <arg>` - (Optional) The maximum number of paths to output when sorted by slack; or maximum number of paths per path group when sorted by group, as specified by `-sort_by`. This is specified as a value greater than or equal to 1. By default the `report_timing` command will report the single worst timing path, or the worst path per path group.

`-nworst <arg>` - (Optional) The number of timing paths per endpoint to output in the timing report. The timing report will return the <N> worst paths based on the specified value, greater than or equal to 1. The default setting is 1.

`-unique_pins` - (Optional) Show only one timing path for each unique set of pins.

`-path_type <arg>` - (Optional) Specify the path data to output in the timing report. The default format is full_clock_expanded. The valid path types are:

- end - Shows the endpoint of the path only, with calculated timing values.

- summary - Displays the startpoints and endpoints with slack calculation.

- short - Displays the startpoints and endpoints with calculated timing values.

- full - Displays the full timing path, including startpoints, through points, and endpoints.

- full_clock - Displays full clock paths in addition to the full timing path.

- full_clock_expanded - Displays full clock paths between a master clock and generated clocks in addition to the full_clock timing path. This is the default setting.

`-input_pins` - (Optional) Show input pins in the timing path report. For use with `-path_type` full, full_clock, and full_clock_expanded.

`-no_header` - (Optional) Do not write a header to the report.

`-no_reused_label` - (Optional) Disable the reporting of reuse information in designs that use incremental place and route based on an existing design checkpoint (DCP) file. Both placement and routing can be completed incrementally, based on prior results stored in a Design Checkpoint file (DCP), using the incremental implementation flow. Refer to the `read_checkpoint` command, or to *Vivado Design Suite User Guide: Implementation (UG904)* for more information on incremental place and route. By default, designs using incremental place and route have pins labeled with information related to the physical data reused from the specified incremental checkpoint. This option removes the reuse labels including the following:

- Routing: Placement and routing to this pin are reused.

- Placement: Cell placement is reused but routing to this pin is not reused.

- Moved: Neither cell placement nor routing to this pin is reused.

- New: The cell, net, or pin is new. It does not exist in the incremental checkpoint.

`-slack_lesser_than <arg>` - (Optional) Report timing on paths with a calculated slack value less than the specified value. Used with `-slack_greater_than` to provide a range of slack values of specific interest.

`-slack_greater_than <arg>` - (Optional) Report timing on paths with a calculated slack value greater than the specified value. Used with `-slack_lesser_than` to provide a range of slack values of specific interest.

`-group` <args> - (Optional) Report timing for paths in the specified path groups. Currently defined path groups can be determined with the `get_path_groups` command.

> 💡 **TIP:** *Each clock creates a path group. Path groups can also be defined with the* `group_path` *command. The* `-group` *option cannot be specified with* `-of_objects`, *which also specifies timing path objects.*

`-sort_by [ slack | group ]` - (Optional) Sort timing paths in the report by slack values, or by path group. Valid values are slack or group. By default, the `report_timing` command reports the worst, or `-nworst`, timing paths in the design. However, with `-sort_by group`, the `report_timing` command returns the worst, or `-nworst`, paths of each path group.

`-no_report_unconstrained` - (Optional) Do not report timing on unconstrained paths. Without this option specified, the `report_timing` command will include unconstrained paths which will have infinite slack.

`-user_ignored` - (Optional) Report only the paths that are usually ignored by timing due to presence of `set_false_path` or `set_clock_groups` constraints.

Note: The `-user_ignored` and `-no_report_unconstrained` options are mutually exclusive and cannot be specified together. The `-user_ignored` option is also mutually exclusive with the `-slack_lesser_than` and `-slack_greater_than` options.

`-of_objects` <args> - (Optional) Report timing on the specified timing path objects. Used with the `get_timing_paths` command.

> 💡 **TIP:** *The `-of_objects` option cannot be used with the various forms of `-from`, `-to`, or `-through` options which are also used to identify timing paths to report. The `-of_objects` option, which defines a timing path object containing a DELAY_TYPE property, cannot be used with `-setup`, `-hold` or `-delay_type`, which all also define a delay type. The `-of_objects` option also cannot be specified with `-group`, which defines groups of timing path objects.*

`-significant_digits` <arg> - (Optional) The number of significant digits in the output results. The valid range is 0 to 3. The default setting is 3 significant digits.

`-column_style [ variable_width | anchor_left | fixed_width ]` - (Optional) Specify the format of the timing path portion of the timing report output. The default format is anchor_left.

`-file` <arg> - (Optional) Write the report into the specified file. The specified file will be overwritten if one already exists, unless `-append` is also specified.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-name` <arg> - (Optional) Specifies the name of the results set for the GUI.

`-no_pr_attribute` <arg> - (Optional) This option disables the standard reporting of the Partial Reconfiguration (PR) attribute data. For PR designs, the logical path is appended to identify cells as belonging to a reconfigurable partition (:RP#), or to the static region of the design (:S).

`-routable_nets` - (Optional) Report the number of routable nets traversed as a property of the timing paths returned.

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-warn_on_violation` - (Optional) Specify that a Critical Warning will be generated by the Vivado Design Suite when the timing report contains a timing violation.

`-cells` <arg> - (Option) Generate the timing report on the specified hierarchical cells. The details of the report will be based on the specified cells rather than the whole design.

-rpx <arg> - (Optional) Specify the file name and path of an Xilinx report file (RPX) to write. This is different from writing the report results to a file using the -file argument. The RPX file is an interactive report that contains all the report information and can be reloaded into memory in the Vivado Design Suite using the open_report command. You should add a .rpx file extension to the specified file name, as the Vivado tool will not automatically assign a file extension.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the set_msg_config command.

Examples

The following example reports the timing for the 5 worst paths in the design, reporting the full timing path, including input pins, with timing values:

```
report_timing -nworst 5 -path_type full -input_pins
```

The following example shows the use of the multiple through points to define both a specific path (through state_reg1) and alternate paths (through count_3 or count_4), and writes the timing results to the specified file:

```
report_timing -from go -through {state_reg1} \
    -through { count_3   count_4 } \
    -to done -path_type summary -file C:/Data/timing1.txt
```

See Also

- get_path_groups
- get_timing_paths
- group_path
- place_design
- report_timing_summary
- route_design
- set_clock_groups
- set_false_path
- get_msg_config

Send Feedback

- set_msg_config

report_timing_summary

Report timing summary

Syntax

```
report_timing_summary [-check_timing_verbose] [-delay_type <arg>]
    [-no_detailed_paths] [-setup] [-hold] [-max_paths <arg>]
    [-nworst <arg>] [-unique_pins] [-path_type <arg>] [-no_reused_label]
    [-input_pins] [-no_pr_attribute] [-routable_nets]
    [-slack_lesser_than <arg>] [-report_unconstrained]
    [-significant_digits <arg>] [-no_header] [-file <arg>] [-append]
    [-name <arg>] [-return_string] [-warn_on_violation] [-datasheet]
    [-cells <args>] [-rpx <arg>] [-quiet] [-verbose]
```

Usage

Name	Description
[-check_timing_verbose]	produce a verbose report when checking the design for potential timing problems
[-delay_type]	Type of path delay: Values: max, min, min_max Default: min_max
[-no_detailed_paths]	do not report timing paths for each clock and path group analyzed
[-setup]	Report max delay timing paths (equivalent to -delay_type max)
[-hold]	Report min delay timing paths (equivalent to -delay_type min)
[-max_paths]	Maximum number of paths to report per clock or path group: Value >=1 Default: 1
[-nworst]	List up to N worst paths to endpoint: Value >=1 Default: 1
[-unique_pins]	for each unique set of pins, show at most 1 path per path group
[-path_type]	Format for path report: Values: end summary short full full_clock full_clock_expanded Default: full_clock_expanded
[-no_reused_label]	Do not label reuse status on pins in the report
[-input_pins]	Show input pins in path
[-no_pr_attribute]	for Dynamic Function eXchange designs, do not report whether netlist resources are in the static or reconfigurable regions
[-routable_nets]	store the number of routable nets traversed as a property on timing paths.
[-slack_lesser_than]	Display paths with slack less than this Default: 1e+30
[-report_unconstrained]	report unconstrained and user ignored paths
[-significant_digits]	Number of digits to display: Range: 0 to 3 Default: 3
[-no_header]	do not generate a report header
[-file]	Filename to output results to. (send output to console if -file is not used)
[-append]	Append the results to file, don't overwrite the results file

Send Feedback

Name	Description
[-name]	Output the results to GUI panel with this name
[-return_string]	return report as string
[-warn_on_violation]	issue a critical warning when the report contains a timing violation
[-datasheet]	Include data sheet report
[-cells]	run report_timing_summary on the specified cell(s)
[-rpx]	Filename to output interactive results to.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Report, Timing

Description

> **TIP:** *The* `report_timing_summary` *can be multi-threaded to speed the process. Refer to the* `set_param` *command for more information on setting the* `general.maxThreads` *parameter.*

Generate a timing summary to help understand if the design has met timing requirements. The timing summary can be run on an open Synthesized or Implemented Design.

The timing summary report includes the following information:

- Timer Settings - Details the timing engine settings used to generate the timing information in the report.

- Check Timing - Contains the same information that is produced by the `check_timing` command, which summarizes potential timing issues.

- Design Timing Summary - Provides a summary of the timing of the design, including values for worst and total negative slack (WNS/TNS), worst and total hold slack (WHS/THS), and component switching limits (CSL).

- Clock Definitions - Contains the same information that is produced by the `report_clocks` command, showing all the clocks that were created for the design, either by `create_clock`, `create_generated_clock`, or automatically by the tool.

- Intra-Clock Table - Summarizes timing paths with the same source and destination clocks.

- Inter-Clock Table - Summarizes timing paths with different source and destination clocks.

- Path Group Table - Shows default path groups and user-defined path groups created by the `group_path` command.

- Timing Details - Contains detailed timing paths, both max delay and min delay, as well as component switching limits for each clock defined, similar to the `report_timing` command.

- Data sheet - Contains the same information that is produced by the `report_datasheet` command. It contains the timing characteristics of a design at the I/O ports. The data sheet information is added to the summary report only when the `-datasheet` option is specified.

This command is automatically run during implementation as part of the `launch_runs` command.

Note: By default the report is written to the Tcl console or STD output. However, the results can also be written to a file or returned as a string if desired.

Arguments

`-check_timing_verbose` - (Optional) Output a verbose timing summary report.

`-delay_type <arg>` - (Optional) Specifies the type of delay to analyze when running the timing report. The valid values are min, max, min_max. The default setting for `-delay_type` is min_max.

`-no_detailed_paths` - (Optional) Do not report the full timing path for each clock or path group analyzed.

`-setup` - (Optional) Check for setup violations. This is the same as specifying `-delay_type max`.

`-hold` - (Optional) Check for hold violations. This is the same as specifying `-delay_type min`.

Note: `-setup` and `-hold` can be specified together, which is the same as specifying `-delay_type min_max`.

`-max_paths <arg>` - (Optional) The maximum number of paths to report per clock or per path group. This is specified as a value greater than or equal to 1. By default the `report_timing_summary` command will report the single worst timing path, or the worst path per path group.

`-nworst <arg>` - (Optional) The number of timing paths to output in the timing report. The timing report will return the <N> worst paths to endpoints based on the specified value, greater than or equal to 1. The default setting is 1.

`-unique_pins` - (Optional) Only report timing paths through each unique set of pins, reporting one path per path group.

`-path_type <arg>` - (Optional) Specify the path data to output in the timing summary report. The default format is full_clock_expanded. The valid path types are:

- `end` - Shows the endpoint of the path only, with calculated timing values.

- `summary` - Displays the startpoints and endpoints with slack calculation.

- `short` - Displays the startpoints and endpoints with calculated timing values.

Send Feedback

- `full` - Displays the full timing path, including startpoints, through points, and endpoints.

- `full_clock` - Displays full clock paths in addition to the full timing path.

- `full_clock_expanded` - Displays full clock paths between a master clock and generated clocks in addition to the full_clock timing path. This is the default setting.

`-no_reused_label` - (Optional) Disable the reporting of reuse information in designs that use incremental place and route based on an existing design checkpoint (DCP) file. Both placement and routing can be completed incrementally, based on prior results stored in a Design Checkpoint file (DCP), using the incremental implementation flow. Refer to the `read_checkpoint` command, or to *Vivado Design Suite User Guide: Implementation (UG904)* for more information on incremental place and route. By default, designs using incremental place and route have pins labeled with information related to the physical data reused from the specified incremental checkpoint. This option removes the reuse labels including the following:

- Routing: Placement and routing to this pin are reused.

- Placement: Cell placement is reused but routing to this pin is not reused.

- Moved: Neither cell placement nor routing to this pin is reused.

- New: The cell, net, or pin is new. It does not exist in the incremental checkpoint.

`-input_pins` - (Optional) Show input pins in the timing path report. For use with `-path_type` full, full_clock, and full_clock_expanded.

`-no_pr_attribute <arg>` - (Optional) For Partial Reconfiguration (PR) designs, this option disables the standard reporting of data that indicates whether netlist resources are in the static or reconfigurable regions.

`-routable_nets` - (Optional) Report the number of routable nets traversed as a property of the timing paths returned.

`-slack_lesser_than <arg>` - (Optional) Report timing on paths with a calculated slack value less than the specified value.

`-report_unconstrained` - (Optional) Report timing on unconstrained paths in the current design. As a default, the `report_timing_summary` command will not include unconstrained timing paths.

> 💡 **TIP:** *In the Vivado IDE, this option must be enabled to report the **User Ignored Paths** in the Timing report.*

`-significant_digits <arg>` - (Optional) The number of significant digits in the output results. The valid range is 0 to 3. The default setting is 3 significant digits.

`-no_header` - (Optional) Do not add header information to the report. This can be useful when returning the timing summary report to a string for further processing.

`-file <arg>` - (Optional) Write the report into the specified file. The specified file will be overwritten if one already exists, unless `-append` is also specified.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-name <arg>` - (Optional) Specifies the name of the results set for the GUI. Timing summary reports in the GUI can be deleted by the `delete_timing_results` command.

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-warn_on_violation` - (Optional) Specify that a Critical Warning will be generated by the Vivado Design Suite when the timing report contains a timing violation.

`-datasheet` - (Optional) Generate data sheet information to add to the summary report.

`-cells <arg>` - (Option) Generate the timing summary report on the specified hierarchical cells. The details of the report will be based on the specified cells rather than the whole design.

`-rpx <arg>` - (Optional) Specify the file name and path of an Xilinx report file (RPX) to write. This is different from writing the report results to a file using the `-file` argument. The RPX file is an interactive report that contains all the report information and can be reloaded into memory in the Vivado Design Suite using the `open_report` command. You should add a `.rpx` file extension to the specified file name, as the Vivado tool will not automatically assign a file extension.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example reports the timing summary of the current design:

```
report_timing_summary
```

The following example reports the hold timing summary of the current design, including unconstrained paths, with the specified options:

```
report_timing_summary -delay_type min -path_type full_clock_expanded \
    -report_unconstrained -max_paths 2 -nworst 1 -significant_digits 2 \
    -input_pins -name {timing_6}
```

See Also

- check_timing
- create_clock
- create_generated_clock
- delete_timing_results
- get_path_groups
- get_timing_paths
- group_path
- open_report
- report_clocks
- report_timing
- set_msg_config
- report_datasheet

report_transformed_primitives

Report details of Unisim primitive transformations.

Syntax

```
report_transformed_primitives [-file <arg>] [-append] [-return_string]
    [-quiet] [-verbose]
```

Usage

Name	Description
[-file]	Output file
[-append]	Append the results to file
[-return_string]	return report as string
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Report

Description

Report the transformed primitives in the current design.

As part of the process of opening the Synthesized design, and loading it into memory, the tool will transform legacy netlist primitives to the supported subset of Unisim primitives.

As a default this report will be written to the standard output. However, the report can also be written to a file or returned to a Tcl string variable for further processing.

Arguments

`-file <arg>` - (Optional) Write the transformed primitives report into the specified file. The specified file will be overwritten if one already exists, unless `-append` is also specified.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

Send Feedback

`-return_string` - (Optional) Directs the output to a Tcl string. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example reports the transformed primitives in the current design, and returns the result to the specified Tcl variable:

```
set transPrim [ report_transformed_primitives -return_string ]
```

report_utilization

Compute utilization of device and display report

Syntax

```
report_utilization [-file <arg>] [-append] [-pblocks <args>]
    [-evaluate_pblock] [-exclude_child_pblocks] [-exclude_non_assigned]
    [-cells <args>] [-return_string] [-slr] [-packthru] [-name <arg>]
    [-no_primitives] [-omit_locs] [-hierarchical] [-spreadsheet_file <arg>]
    [-spreadsheet_table <arg>] [-spreadsheet_depth <arg>]
    [-hierarchical_depth <arg>] [-hierarchical_percentages] [-quiet]
    [-verbose]
```

Returns

Report

Usage

Name	Description
[-file]	Filename to output results to. (send output to console if -file is not used)
[-append]	Append the results to file, don't overwrite the results file
[-pblocks]	Report utilization of given pblock
[-evaluate_pblock]	Report utilization with demand as specified cells and supply as specified pblock area
[-exclude_child_pblocks]	Report utilization with out child pblocks
[-exclude_non_assigned]	Pblock utilization with out Non-assigned Cells
[-cells]	Report utilization of given list of cells
[-return_string]	Return report as string
[-slr]	SLR wise utilization of resources
[-packthru]	Reports LUTs used exclusively as pack-thru
[-name]	Output the results to GUI panel with this name
[-no_primitives]	Removes "Primitives Section" from report_utilization o/p.
[-omit_locs]	Removes "Loced" column from report_utilization o/p.
[-hierarchical]	Generates text-based hierarchical report.
[-spreadsheet_file]	Specify file for exporting utilization tables as spreadsheets. This feature is available only in GUI mode.
[-spreadsheet_table]	Choose a particular utilization table to export as spreadsheet file. Default value : Hierarchy
[-spreadsheet_depth]	Specifies the depth level for spreadsheet. Default value : 8 Default: 8
[-hierarchical_depth]	Specifies the depth level for textual hierarchical report Default: 0
[-hierarchical_percentages]	Report percentages in textual hierarchical report
[-quiet]	Ignore command errors

Send Feedback

Name	Description
`[-verbose]`	Suspend message limits during command execution

Categories

[Report](#)

Description

Report resource usage on the target part by the current synthesized or implemented design. The report is returned to the standard output, unless the `-file`, `-return_string`, or `-name` arguments are specified.

> 💡 **TIP:** *Though resource usage can be reported early in the design process, the report will be more accurate as the design progresses from synthesis through implementation.*

This command returns the requested information, or returns an error if it fails.

Arguments

`-file <arg>` - (Optional) Write the report into the specified file. The specified file will be overwritten if one already exists, unless `-append` is also specified.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-pblocks <arg>` - (Optional) Report the resources utilized by one or more Pblocks in the design.

Note: `-pblocks` can not be used with the `-name` option.

`-evaluate_pblock <arg>` - (Optional) Performs a what-if analysis on the specified `-pblocks` and the specified `-cells`. This option reports the utilization of the Pblocks as if they were populated by the specified cells. Percentages of utilization represent the ratio of the assigned cells to the available resources. This option requires both `-pblocks` and `-cells` to be specified.

`-exclude_child_pblocks` - (Optional) Report utilization of the specified `-pblocks` excluding any nested Pblocks. This has the effect of reducing the utilization of the region as it does not account for cells that are assigned to a Pblock through a nested Pblock.

`-exclude_non_assigned` - (Optional) Exclude non-assigned cells from the Utilization report when the `-pblocks` option is specified. This has the effect of decreasing the utilization of the Pblock region as it does not account for cells that are utilized, but are not assigned to the specified Pblocks.

`-cells <arg>` - (Optional) Report the resources utilized by one or more hierarchical cells in the current design.

`-return_string` - (Optional) Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-slr` - (Optional) Reports the utilization for each separate SLR in devices having SLRs.

`-packthru` - (Optional) Reports LUTs used for route through purposes. This appears in the utilization report as "LUTs used exclusively as route-thrus".

`-name <arg>` - (Optional) Specifies the name of the results to output to the GUI.

`-no_primitives` - (Optional) Remove the Primitives section from the report. The Primitives section reports the number and type of logic primitives used on the device.

`-omit_locs` - (Optional) Omit the `Fixed` column from the report. The `Fixed` column reports the quantity of logic elements that the user has fixed, or manually placed onto the device.

`-hierarchical` - (Optional) Reports the utilization of the device broken down according to the hierarchy of the design.

`-hierarchical_depth <arg>` - (Optional) Specifies the depth of the hierarchy to report when reporting utilization according to the hierarchy. The default depth is 0, which means that `-hierarchical` will report full design hierarchy.

`-hierarchical_percentages` - (Optional) Specifies that utilization data in the hierarchical report be reported as percentages.

`-spreadsheet_file <arg>` - (Optional) Export utilization tables to the specified XLSX format spreadsheet. The ability to export a spreadsheet file is only available when the `-name` option is also specified and the report is generated in the GUI.

> **TIP:** *You should specify the* `.xlsx` *suffix for the specified file, as it is not automatically assigned. If the path is not specified as part of the file name, the file will be written into the current working directory, which may be the directory from which the Vivado tool was launched.*

`-spreadsheet_table <arg>` - (Optional) Specify a utilization table to export to the spreadsheet file. The default is to export the Hierarchy table for the whole design. This option requires the use of `-spreadsheet_file`. The table name is displayed in the Report Utilization window when a specific table is selected from the tree view. Some example table names are:

- "Hierarchy"

- "Slice Logic Distribution"

- "Slice Logic Distribution - LUT as Memory"

- "Slice Logic Distribution - LUT as Memory - LUT as Distributed RAM"

- "Slice Logic Distribution - LUT as Memory - LUT as Shift Register"

- "Slice Logic Distribution - LUT as Logic"

- "Slice Logic - F8 Muxes"

- "Slice Logic - Slice Registers"

- "Slice Logic - Slice Registers - Registers as AND/OR"

- "Memory - Block RAM Tile"

- "Memory - Block RAM Tile - RAMB18"

- "DSP - DSPs"

- "Clocking - BUFGCTRL"

- "Specific Feature - BSCANE2"

- "Primitives"

`-spreadsheet_depth <arg>` - (Optional) Specifies the hierarchical depth, starting from the top-level of the design, to export to the spreadsheet file. The default value is 8. This options requires the use of `-spreadsheet_file`.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example reports the resources collectively utilized by all the Pblocks in the design, and writes the results to the specified file:

```
report_utilization -pblocks [get_pblocks] -file C:/Data/pblocks_util.txt
```

This example reports the utilization for the whole design to the named report in the GUI, but exports the "Clocking - BUFGCTRL" table to the specified spreadsheet file:

```
report_utilization -name utilization_1 -spreadsheet_file util_table.xlsx \
-spreadsheet_table "Clocking - BUFGCTRL"
```

The following example reports the resources utilized by each Pblock in the design, appending the report for each Pblock to a single specified file:

```
foreach x [get_pblocks] {
  puts "Reporting Pblock: $x ----------------------------------"
  report_utilization -append -file C:/Data/pblocks_util.txt -pblocks $x
}
```

See Also

- delete_utilization_results

report_values

Print current simulated value of given HDL objects (variable, signal, wire, or reg)

Syntax

```
report_values [-radix <arg>] [-quiet] [-verbose] [<hdl_objects>...]
```

Returns

Print name and value of HDL objects on the console in textual format.

Usage

Name	Description
[-radix]	The radix specifies the radix to use for printing the values of the hdl_objects. Allowed values are: default, dec, bin, oct, hex, unsigned, ascii, smag.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<hdl_objects>]	The hdl_objects to report. Default is report_objects [get_objects *]

Categories

Simulation

Description

Report the values of the specified HDL objects at the current simulation run time.

HDL objects include HDL signals, variables, or constants as defined in the Verilog or VHDL test bench and source files. An HDL signal includes Verilog wire or reg entities, and VHDL signals. Examples of HDL variables include Verilog real, realtime, time, and event.

HDL constants include Verilog parameters and localparams, and VHDL generic and constants. The HDL scope, or scope, is defined by a declarative region in the HDL code such as a module, function, task, process, or begin-end blocks in Verilog. VHDL scopes include entity/architecture definitions, block, function, procedure, and process blocks.

Arguments

`-radix <arg>` - (Optional) Specifies the radix to use when returning the value of the specified objects. Allowed values are: `default`, `dec`, `bin`, `oct`, `hex`, `unsigned`, `ascii`, or `smag`.

Send Feedback

Note: The radix `dec` indicates a signed decimal. Specify the radix `unsigned` when dealing with unsigned data.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hdl_objects> - (Required) Specifies one or more HDL objects to return the values of. The object can be specified by name, or can be returned as an object from the `get_objects` command.

Examples

The following example reports the value of all objects at the current time:

```
report_values [get_objects]
```

This example shows the difference between the `bin`, `dec`, and `unsigned` radix on the value returned from the specified bus:

```
report_values -radix bin /test/bench_VStatus_pad_0_i[7:0]
  Declared: {/test/bench_VStatus_pad_0_i[7:0]} Verilog 10100101
report_values -radix unsigned /test/bench_VStatus_pad_0_i[7:0]
  Declared: {/test/bench_VStatus_pad_0_i[7:0]} Verilog 165
report_values -radix dec /test/bench_VStatus_pad_0_i[7:0]
  Declared: {/test/bench_VStatus_pad_0_i[7:0]} Verilog -91
```

See Also

- current_time
- get_objects
- get_value
- set_value

report_waivers

Report status of DRC/METHODOLOGY/CDC message waivers

Syntax

```
report_waivers [-file <arg>] [-type <arg>] [-write_valid_waivers]
    [-write_ignored_waivers] [-append] [-return_string]
    [-show_msgs_with_no_waivers] [-quiet] [-verbose]
```

Usage

Name	Description
[-file]	Name of file to report waivers
[-type]	Type of waiver - ALL, DRC, METHODOLOGY, CDC
[-write_valid_waivers]	(special) Specifies writing out the specific waivers which were used in the last report_drc/methodology/cdc run(s)
[-write_ignored_waivers]	(special) Specifies writing out the specific waivers which were NOT used in the last report_drc/methodology/cdc run(s)
[-append]	Append the current report results to the file specified with -file
[-return_string]	Return report results as a string object
[-show_msgs_with_no_waivers]	also list report_drc/methodology/cdc messages which have no defined waivers
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Waiver, Report, Object

Description

Reports DRC, METHODOLOGY, and CDC violation messages and displays what waivers are in place in the current design.

In addition, the `report_drc`, `report_methodology`, and `report_cdc` commands have options to run the reports on waived violations or checks.

Arguments

`-file <arg>` - (Optional) Write the waivers report into the specified file. The specified file will be overwritten if one already exists.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

Send Feedback

`-type <arg>` - (Optional) Specifies the type of waiver to report. Currently supports DRC, METHODOLOGY, and CDC. If the `-type` is not specified, all waivers will be reported.

`-write_valid_waivers <arg>` - (Optional) This option requires the use of the `-file` option, and writes the valid waivers that were used during the last run of the DRC, methodology, or CDC reports.

`-write_ignored_waivers <arg>` - (Optional) This option requires the use of the `-file` option, and writes the waivers that were NOT used during the last run of the DRC, methodology, or CDC reports.

`-append` - (Optional) Append the output of the report to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-return_string` - (Optional) Directs the report output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note: This argument cannot be used with the `-file` option.

`-show_msgs_with_no_waivers` - (Optional) By default the `report_waivers` command will hide data for all DRC, METHODOLOGY, and CDC violations, which do not have waivers defined. This options restores the reporting of violation messages which have no defined waivers.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

This example reports all waivers in the current design:

```
report_waivers
```

See Also

- create_waiver
- delete_waivers
- get_waivers

- report_cdc
- report_drc
- report_methodology
- write_waivers

Send Feedback

reset_drc

Remove DRC report

Syntax

```
reset_drc [-name <arg>] [-quiet] [-verbose]
```

Usage

Name	Description
[-name]	DRC result name
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

DRC, Report

Description

Clear the DRC results from the specified named result set.

This command operates silently, returning nothing if successful, or returning an error if it fails.

Arguments

`-name <arg>` - (Optional) Specifies the name of the DRC results to be cleared. The name is established by the `-name` argument in the `report_drc` command.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example clears the specified results set from memory and the GUI:

```
reset_drc -name DRC1
```

Send Feedback

See Also

- report_drc

reset_drc_check

Reset one or more DRC checks to factory defaults.

Syntax

```
reset_drc_check [-quiet] [-verbose] [<checks>...]
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<checks>]	The list of checks to reset.

Categories

DRC, Object

Description

Reset the specified DRC checks to the defaults provided by the Vivado Design Suite. This will restore the DRC check to its default configuration, including any changes to the IS_ENABLED or SEVERITY properties.

The IS_ENABLED property can be modified on a specific DRC check to disable the rule from being checked, even when it is specified either directly in the `report_drc` command, or as part of a ruledeck.

The SEVERITY property is a string property that can be modified to change the severity associated with a specific DRC rule when a violation is found during the `report_drc` command. The supported values are: FATAL, ERROR, "CRITICAL WARNING", WARNING, ADVISORY

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Send Feedback

<checks> - (Required) The list of one or more DRC rule checks to reset to the tool defaults.

Examples

The following example modifies the IS_ENABLED property for the ROAS-1 rule, modifies the SEVERITY property for the RFFC-1 rule, and then restores the default settings for all checks:

```
set_property IS_ENABLED false [get_drc_checks ROAS-1]
set_property SEVERITY "Critical Warning" [get_drc_checks RFFC-1]
reset_drc_check [get_drc_checks]
```

See Also

- add_drc_checks
- get_drc_checks
- report_drc
- set_property

reset_hw_axi

Reset hardware AXI core state.

Syntax

```
reset_hw_axi [-quiet] [-verbose] [<hw_axis>...]
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<hw_axis>]	List of hardware AXI objects.

Categories

Hardware

Description

Reset the STATUS properties of the specified hw_axi objects, or the current device.

The `reset_hw_axi` restores the hw_axi core on the current device to a known state from which to begin running AXI transactions. The STATUS properties include:

- STATUS.AXI_READ_BUSY
- STATUS.AXI_READ_DONE
- STATUS.AXI_WRITE_BUSY
- STATUS.AXI_WRITE_DONE
- STATUS.BRESP - Write Response Channel Response. Indicates results of the write transfer.
- STATUS.RRESP - Read Response Channel Response. Indicates results of the read transfer.

The command returns nothing if successful, and returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

Send Feedback

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_axis> - (Required) The hw_axi objects to reset. The hw_axi must be specified as an object returned by the `get_hw_axis` command.

Example

The following example resets the hw_axis on the current device, restoring initial settings:

```
reset_hw_axis [get_hw_axis]
```

See Also

- create_hw_axi_txn
- delete_hw_axi_txn
- get_hw_axis
- get_hw_axi_txns
- refresh_hw_axi

reset_hw_ila

Reset hardware ILA control properties to default values.

Syntax

```
reset_hw_ila [-reset_compare_values <arg>] [-quiet] [-verbose]
    [<hw_ilas>...]
```

Usage

Name	Description
[-reset_compare_values]	Reset associated hardware probe compare values.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<hw_ilas>]	List of hardware ILA objects. Default: Current hardware ILA

Categories

Hardware

Description

Reset the trigger and capture configuration properties on the specified ILA debug core, and the TRIGGER_COMPARE_VALUE and CAPTURE_COMPARE_VALUE properties on the core's debug probes.

Properties of the hw_ila object are configured with the `set_property` command in preparation for the `run_hw_ila` command to configure the ILA core on the hw_device. This command restores the user-configurable properties on the specified hw_ila to their default settings. Refer to the *Vivado Design Suite User Guide: Programming and Debugging (UG908)* for more information on these properties.

The default properties are:

- CONTROL.DATA_DEPTH is set to the MAX_DATA_DEPTH of the hw_ila object.

- CONTROL.TRIGGER_POSITION 0

- CONTROL.WINDOW_COUNT 1

- CONTROL.TRIGGER_MODE BASIC_ONLY

- CONTROL.TRIGGER_CONDITION AND

- CONTROL.TRIG_OUT_MODE DISABLED

- CONTROL.CAPTURE_MODE ALWAYS

Send Feedback

- CONTROL.CAPTURE_CONDITION AND

- TRIGGER_COMPARE_VALUE eq1'bX (on the hw_probes)

- CAPTURE_COMPARE_VALUE eq1'bX (on the hw_probes)

This command operates silently, returning nothing if successful, or returning an error if it fails.

Arguments

`-reset_compare_values [ true | false ]` - (Optional) Reset the TRIGGER_COMPARE_VALUE and CAPTURE_COMPARE_VALUE properties on the hw_probes associated with the specified hw_ila object. This is a boolean argument that is TRUE, or enabled, by default. If `-reset_compare_values false` is used, the compare value properties on the probes are not reset. In this case, the properties on the hw_ila are reset, but the properties on the hw_probes are left as currently configured.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_ilas> - (Optional) Specify one or more hw_ila objects to reset. The hw_ila objects can either be specified as objects returned by the `get_hw_ilas` or `current_hw_ila` commands, or specified by name. If the hw_ila is not specified, the `current_hw_ila` will be reset.

Example

The following example resets all hw_ila debug cores on the current device:

```
reset_hw_ila [get_hw_ilas]
```

See Also

- current_hw_ila
- get_hw_ilas
- run_hw_ila
- set_property

reset_hw_pcie

Reset the PCIe core.

Syntax

```
reset_hw_pcie [-quiet] [-verbose] <hw_pcie>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hw_pcie>	Hardware PCIe object

Categories

[Hardware](#)

Description

Reset the Peripheral Component Interconnect Express (PCIe) debug core object, hw_pcie, defined on the current hardware device.

The customizable LogiCORE IP PCIe core for Xilinx ACAPs is designed for evaluating and monitoring the PCIe Link Training and Status State Machine (LTSSM) running on the Gigabit Transceivers (GTs). In the Vivado Hardware Manager, Versal PCIe soft cores implemented in the design, are represented as hw_pcie objects. You can use these PCIe debug cores to solve a range of debug and validation problems; from viewing the PCIe link info to the LTSSM state transition diagram.

This command resets the PCIe soft IP in hardware.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the set_msg_config command.

`<hw_pcie>` - (Required) List of HW PCIe objects.

Examples

The following example resets PCIe core at index 0

```
reset_hw_pcie [lindex [get_hw_pcies] 0]
```

See Also

- get_hw_pcies
- report_hw_pcie
- refresh_hw_pcie

reset_hw_vio_activity

Reset hardware VIO ACTIVITY_VALUE properties, for hardware probes associated with specified hardware VIO objects.

Syntax

```
reset_hw_vio_activity [-quiet] [-verbose] <hw_vios>...
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hw_vios>	List of hardware VIO objects.

Categories

Hardware

Description

Resets the ACTIVITY_VALUE properties for all hardware probes on the specified VIO debug core objects. The ACTIVITY_VALUE property is used by the Vivado IDE to represent transitions on the input probes of the VIO debug cores.

In addition to reading values from the VIO input probes, you can also monitor the activity of the VIO input probes. The ACTIVITY_VALUE property is used to indicate when the values on the VIO inputs have changed in between periodic updates to the Vivado IDE. Refer to the *Vivado Design Suite User Guide: Programming and Debugging (UG908)* for more information.

This command returns nothing if successful, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_vios> - (Required) Specify one or more hw_vio objects to reset. The hw_vio objects can either be specified as objects returned by the `get_hw_vios` command, or specified by name.

Example

The following example resets the input activity properties of the VIO debug core:

```
reset_hw_vio_activity [get_hw_vios]
```

See Also

- commit_hw_vio
- connect_hw_server
- current_hw_device
- get_hw_probes
- get_hw_vios
- program_hw_devices
- refresh_hw_vio
- reset_hw_vio_outputs
- set_property

reset_hw_vio_outputs

Reset hardware VIO core outputs to initial values.

Syntax

```
reset_hw_vio_outputs [-quiet] [-verbose] <hw_vios>...
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hw_vios>	List of hardware VIO objects.

Categories

[Hardware](#)

Description

Reset the hardware VIO debug core outputs to their initial, or "reset" state.

The Virtual Input/Output (VIO) debug core can both monitor and drive internal signals on a programmed Xilinx FPGA in real time. The VIO core uses hardware probes, hw_probe objects, to monitor and drive signals on the device. Input probes monitor signals as inputs to the VIO core. Output probes drive signals to specified values from the VIO core.

The `reset_hw_vio_outputs` command restores the signal values at the output probes of the specified hw_vio debug cores to their initial values. This affects the signal on the hw_device, but does not affect the OUTPUT_VALUE property of the hw_probe objects.

> 💡 **TIP:** *This command has the effect of resetting the initial value of the signal on the hw_vio debug core, without resetting the properties on the hw_probe object.*

This command returns nothing if successful, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the set_msg_config command.

<hw_vios> - (Required) Specify one or more hw_vio objects to reset. The hw_vio objects can either be specified as objects returned by the get_hw_vios command, or specified by name.

Example

The following example resets the output probes on the VIO debug core to the initial values on the core when the FPGA was first configured and booted:

```
reset_hw_vio_outputs [get_hw_vios {hw_vio_1}]
```

See Also

- commit_hw_vio
- connect_hw_server
- current_hw_device
- get_hw_probes
- get_hw_vios
- program_hw_devices
- refresh_hw_vio
- reset_hw_vio_activity
- set_property

Send Feedback

reset_methodology

Remove Methodology report

Syntax

```
reset_methodology [-name <arg>] [-quiet] [-verbose]
```

Usage

Name	Description
[-name]	Methodology result name
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Methodology](), [Report]()

Description

Clear the methodology results from the specified named result set.

This command operates silently, returning nothing if successful, or returning an error if it fails.

Arguments

`-name <arg>` - (Optional) Specifies the name of the methodology results to be cleared. The name is established by the `-name` argument in the `report_methodology` command.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example clears the specified results set from memory and the GUI:

```
reset_methodology -name ultrafast_methodology_3
```

Send Feedback

See Also

- get_methodology_violations
- report_methodology
- reset_methodology_check

Send Feedback

reset_methodology_check

Reset one or more Methodology checks to factory defaults.

Syntax

```
reset_methodology_check [-quiet] [-verbose] [<checks>...]
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<checks>]	The list of checks to reset.

Categories

Methodology, Object

Description

Reset the specified methodology checks to the defaults provided by the Vivado Design Suite. This will restore the check to its default configuration, including any changes to the IS_ENABLED or SEVERITY properties.

The IS_ENABLED property can be modified on a specific methodology check to disable the rule from being checked, even when it is specified directly in the `report_methodology` command.

The SEVERITY property is an enumerated property that can be modified to change the severity associated with a specific methodology check when a violation is found during the `report_methodology` command. The supported values are: FATAL, ERROR, "CRITICAL WARNING", WARNING, ADVISORY

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Send Feedback

<checks> - (Required) The list of one or more DRC rule checks to reset to the tool defaults.

Examples

The following example modifies the IS_ENABLED and SEVERITY properties for the CHECK-4 methodology check, reports the properties of the check to see the changes, and then resets the methodology check to its default setting:

```
set_property IS_ENABLED false [get_methodology_checks CHECK-4]
set_property SEVERITY Warning [get_methodology_checks CHECK-4]
report_property [get_methodology_checks CHECK-4]
reset_methodology_check [get_methodology_checks CHECK-4]
report_property [get_methodology_checks CHECK-4]
```

See Also

- get_methodology_checks
- report_methodology
- reset_methodology

reset_msg_config

Resets or removes a message control rule previously defined by the set_msg_config command.

Syntax

```
reset_msg_config [-string <args>] [-id <arg>] [-severity <arg>] [-limit]
    [-suppress] [-count] [-default_severity] [-regexp] [-quiet] [-verbose]
```

Usage

Name	Description
[-string]	A qualifier, only a rule created with a matching string qualifier will be reset/removed Default: empty
[-id]	A qualifier, only a rule created with a matching id qualifier will be reset/removed
[-severity]	A qualifier, only a rule created with a matching severity qualifier will be reset/removed
[-limit]	reset the limit values for message controls that match the given qualifiers for the current project
[-suppress]	stop suppressing messages that match the given qualifiers for the current project
[-count]	reset the count of messages for all message controls that match the given qualifiers for the current project. This will prevent messages from being suppressed by a -limit control until the message count once again exceeds the specified limit.
[-default_severity]	reset the message severity of all messages controls for the current project that match the given qualifiers to their default value
[-regexp]	The values used for -string are full regular expressions
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Report

Description

This command restores the default settings of the message limits or severity for messages returned by the Vivado tool, or can unsuppress previously suppressed messages, as configured by the set_msg_config command.

You can only perform one reset action for each reset_msg_config command. An error is returned if more than one action is attempted in a single reset_msg_config command.

Send Feedback

Message qualifiers of string, ID, and severity are used to determine which messages are reset by the `reset_msg_config` command. Multiple qualifiers have an AND relationship; only the messages matching the qualifiers will be reset.

Note: You must supply at least one message qualifier to identify a message or group of messages to apply the command to, or an error is returned.

To report the current rule configurations for messages, use the `get_msg_config` command.

Arguments

`-string` <args> - (Optional) Apply the selected operation only to messages that contain the given list of strings. Strings must be enclosed in braces, and multiple strings can be specified separated by spaces:

```
{{Vivado} {Synthesis}}
```

Note: Strings are case sensitive.

`-id <arg>` - (Optional) Reset messages matching the specified message ID. The message ID is included in all returned messages. For example, "Common 17-54" and "Netlist 29-28".

Note: A wildcard * indicates all message IDs should be reset.

`-severity <arg>` - Reset messages with the specified message severity. There are five message severities:

- `ERROR` - An ERROR condition implies an issue has been encountered which will render design results unusable and cannot be resolved without user intervention.

- `{CRITICAL WARNING}` - A CRITICAL WARNING message indicates that certain input/ constraints will either not be applied or are outside the best practices for a FPGA family. User action is strongly recommended.

  Note: Since this is a two word value, it must be enclosed in {}.

- `WARNING` - A WARNING message indicates that design results may be sub-optimal because constraints or specifications may not be applied as intended. User action may be taken or may be reserved.

- `INFO` - An INFO message is the same as a STATUS message, but includes a severity and message ID tag. An INFO message includes a message ID to allow further investigation through answer records if needed.

- `STATUS` - A STATUS message communicates general status of the process and feedback to the user regarding design processing. A STATUS message does not include a message ID.

`-limit` - (Optional) Reset the message limit for messages matching the string, ID, or severity qualifiers.

Send Feedback

`-suppress` - (Optional) Reset, or unsuppress messages matching the string, ID, or severity qualifiers.

`-count` - (Optional) Reset the message count for messages matching the string, ID, or severity qualifiers.

`-default_severity` - (Optional) Restore the default message severity for messages matching the string, ID, or severity qualifiers.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example changes the severity of the specified message ID to a Critical Warning, and then resets the message to its default severity:

```
set_msg_config -id "Common 17-81" -new_severity "CRITICAL WARNING"
reset_msg_config -id "Common 17-81" -default_severity
```

This example changes the severity of messages with the specified message ID, gets the current message configuration rules, and then shows two different command forms to reset the specific rule and restore the message:

```
set_msg_config -id "Common 17-361" -severity INFO -new_severity WARNING
get_msg_config -rules
--------------------
Message control rules currently in effect are:
Rule Name   Rule                                              Current
Message Count
1   set_msg_config -ruleid {1} -id {Common 17-361} -severity {INFO} -
new_severity {WARNING} 0
--------------------
reset_msg_config -id "Common 17-361" -default_severity
reset_msg_config -ruleid {1}
```

> 💡 **TIP:** *In the preceding example, only one of the `reset_msg_config` commands is needed to reset the message.*

See Also

- [get_msg_config](#)

- set_msg_config

Send Feedback

reset_msg_count

Reset message count

Syntax

```
reset_msg_count [-quiet] [-verbose] <id>
```

Returns

New message count.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<id>	Unique message Id to be reset, e.g. "Common 17-99". "reset_msg_count -id *" reset all counters

Categories

[Report](#)

Description

Reset the message count for the specified message ID to 0. This restarts the message counter toward the specified message limit. This can be used to reset the count of specific messages that may be reaching the limit, or reset the count of all messages returned by the tool.

Every message delivered by the tool has a unique global message ID that consists of an application sub-system code and a message identifier. This results in a message ID that looks like the following:

```
"Common 17-54"
"Netlist 29-28"
"Synth 8-3295"
```

You can get the current message count for a specific message ID using the `get_msg_count` command.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<id> - (Required) Specifies the message ID to reset the count to 0. Specify * to reset the count of all messages to 0.

Examples

The following example resets the message count for all messages:

```
reset_msg_count *
```

See Also

- set_msg_config

Send Feedback

reset_operating_conditions

Reset operating conditions to tool default for power estimation

Syntax

```
reset_operating_conditions [-voltage <args>] [-grade] [-process]
    [-junction_temp] [-ambient_temp] [-thetaja] [-thetasa] [-airflow]
    [-heatsink] [-thetajb] [-board] [-board_temp] [-board_layers]
    [-design_power_budget] [-supply_current_budget <args>] [-quiet]
    [-verbose]
```

Usage

Name	Description
[-voltage]	Resets voltage value. Supported voltage supplies vary by family.
[-grade]	Resets temperature grade
[-process]	Resets process
[-junction_temp]	Resets Junction Temperature
[-ambient_temp]	Resets Ambient Temperature
[-thetaja]	Resets ThetaJA
[-thetasa]	Resets ThetaSA
[-airflow]	Resets Airflow
[-heatsink]	Resets dimensions of heatsink
[-thetajb]	Resets ThetaJB
[-board]	Resets Board type
[-board_temp]	Resets Board Temperature
[-board_layers]	Resets Board layers
[-design_power_budget]	Design Power Budget (W)
[-supply_current_budget]	Resets list of supply current budget 'name value' pairs. Supported voltage supplies vary by family.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Power](#), [XDC](#)

Description

Resets the specified operating conditions to their default values. If no operating conditions are specified, all operating conditions are reset to their default values.

Operating conditions can be set using the `set_operating_conditions` command. The current values can be determined using the `report_operating_conditions` command. The environmental operating conditions of the device are used for power analysis when running the `report_power` command, but are not used during timing analysis.

Note: This command returns nothing if successful, or returns an error if it fails.

Arguments

`-voltage` **<args>** - (Optional) Reset the voltage supply to the default value. The voltage supply and its default depend on the device family.

`-grade` - (Optional) Reset the temperature grade of the selected device. The default value is "commercial".

`-process` - (Optional) Reset the manufacturing process for the target device. The default process is "typical".

`-junction_temp` - (Optional) Reset the junction temperature for the target device. The default value is "auto".

`-ambient_temp` - (Optional) Reset the ambient temperature of the design. The default setting is "default".

`-thetaja` - (Optional) Reset the Theta-JA thermal resistance. The default setting is "auto".

`-thetasa` - (Optional) Reset the Theta-SA thermal resistance. The default setting is "auto".

`-airflow` - (Optional) Reset the Linear Feet Per Minute (LFM) airflow. The default setting varies by device family.

`-heatsink` - (Optional) Reset the heatsink profile. The default setting is "medium".

`-thetajb` - (Optional) Reset the Theta-JB thermal resistance. The default setting is "auto".

`-board` - (Optional) Reset the board size to be used for modeling. The default value is "medium".

`-board_temp` - (Optional) Reset the board temperature to the default setting.

`-board_layers` - (Optional) Reset the number of board layers to be used for modeling to the default setting of "12to15".

`-design_power_budget` - (Optional) Reset the design power budget to "unspecified".

`-supply_current_budget` **<args>** - (Optional) Reset the supply current budgets to the default value. The voltage supply names depend on the device family.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example resets all the operating conditions for the design to their default setting:

```
reset_operating_conditions
```

The following example resets the junction, ambient, and board temperature for the design to their default settings:

```
reset_operating_conditions -junction_temp -ambient_temp -board_temp
```

The following example resets the voltage supply Vccint to its default value:

```
reset_operating_conditions -voltage Vccint
```

See Also

- report_operating_conditions
- report_power
- set_operating_conditions

reset_param

Reset a parameter

Syntax

```
reset_param [-quiet] [-verbose] <name>
```

Returns

Original value

Usage

Name	Description
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<name>`	Parameter name

Categories

PropertyAndParameter

Description

Restores a user-definable configuration parameter that has been changed with the `set_param` command to its default value.

You can use the `report_param` command to see which parameters are currently defined.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - (Required) The name of a parameter to reset. You can only reset one parameter at a time.

Send Feedback

Examples

The following example restores the tcl.statsThreshold parameter to its default value:

```
reset_param tcl.statsThreshold
```

See Also

- get_param
- list_param
- report_param
- set_param

reset_project

Reset current project

Syntax

```
reset_project [-exclude_runs] [-exclude_ips] [-exclude_sim_runs] [-quiet]
    [-verbose]
```

Usage

Name	Description
[-exclude_runs]	Do not reset runs
[-exclude_ips]	Do not reset ips
[-exclude_sim_runs]	Do not reset simulation runs
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Project

Description

Reset the current project to its starting condition, with source and constraint files, by cleaning out the various output files created during synthesis, simulation, implementation, and `write_bitstream`. Also resets the state of the project to the start of the design flow.

> **TIP:** *Any user-defined Tcl variables that are in the global namespace (i.e. not in a project-specific namespace) are not reset or cleared by this command. Global variables are persistent with the invocation of Vivado and are only cleared when the Vivado Design Suite is closed. You can also use the* `unset` *command to expressly clear a specific Tcl variable.*

Arguments

`-exclude_runs` - (Optional) Exclude the `<project>.runs` folder from the reset process. In this case, the runs folder will be preserved, while the rest of the project data will be removed.

`-exclude_ips` - (Optional) Exclude the `<project>.srcs/sources_1/ip` folder from the reset process. In this case, the IP folder, containing the IP cores and generated targets, will be preserved, while the rest of the project data will be removed.

`-exclude_sim_runs` - (Optional) Exclude the `<project>.sim` folder from the reset process. In this case, the simulation folder will be preserved, while the rest of the project data will be removed.

Send Feedback

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Example

The following example resets the current project, while preserving the simulation run data, and returning all messages regardless of message limits:

```
reset_project -exclude_sim_runs -verbose
```

See Also

- create_project
- current_project

Send Feedback

reset_property

Reset property on object(s)

Syntax

```
reset_property [-quiet] [-verbose] <property_name> <objects>...
```

Returns

The value that was set if success, "" if failure.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<property_name>	Name of property to reset
<objects>	Objects to set properties

Categories

[Object](), [PropertyAndParameter]()

Description

Restores the specified property to its default value on the specified object or objects. If no default is defined for the property, the property is unassigned on the specified object.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the set_msg_config command.

<property_name> - (Required) The name of the property to be reset.

`<objects>` - (Required) One or more objects on which the property will be restored to its default value.

Examples

The following example sets the DOB_REG property on the specified Block RAM, and then resets the property:

```
set_property DOB_REG 1 [get_cells usbEngine1/usbEngineSRAM/snoopyRam_reg_19]
reset_property DOB_REG [get_cells usbEngine1/usbEngineSRAM/snoopyRam_reg_19]
```

See Also

- create_property
- get_cells
- get_property
- list_property
- list_property_value
- report_property
- set_property

Send Feedback

reset_runs

Reset an existing run

Syntax

```
reset_runs [-prev_step] [-from_step <arg>] [-quiet] [-verbose] <runs>
```

Usage

Name	Description
[-prev_step]	Reset last run step
[-from_step]	First Step to reset
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<runs>	Runs to modify

Categories

Project

reset_simulation

Reset an existing simulation run

Syntax

```
reset_simulation [-mode <arg>] [-type <arg>] [-quiet] [-verbose] [<simset>]
```

Usage

Name	Description
[-mode]	Remove generated data for the specified mode. Values: behavioral, post-synthesis, post-implementation Default: behavioral
[-type]	Remove generated data for the specified type. Applicable mode is post-synthesis or post-implementation. Values: functional, timing
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<simset>]	Name of the simulation fileset to reset

Categories

Simulation

Description

Reset the current simulation to its starting condition, by cleaning out the various output files created during compilation and simulation for the specified simulation fileset.

⭐ **IMPORTANT!** *Local files will be removed from the project simulation folders without warning.*

The command returns nothing if successful, or an error if it fails.

Arguments

`-mode [ behavioral | post-synthesis | post-implementation ]` - (Optional) Specify the simulation mode to reset. Valid values include behavioral simulation, post-synthesis, or post implementation simulation. The default mode is `behavioral`.

`-type [ functional | timing ]` - (Optional) Cannot be used with `-mode behavioral`. Specifies functional simulation of just the netlist, or timing simulation of the netlist and SDF file. Post-synthesis timing simulation uses SDF component delays from the `synth_design` command. Post-implementation timing simulation uses SDF delays from the `place_design` and `route_design` commands.

Send Feedback

Note: Do not use `-type` with `-mode behavioral`, or the tool will return an error.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<simset> - (Optional) Specify a simulation fileset to remove. The default simset is sim_1.

Examples

The following example resets the post-synthesis timing simulation by removing files for the sim_2 simset:

```
reset_simulation -mode post-synthesis -type timing sim_2
```

See Also

- launch_simulation

reset_ssn

Clear a SSN results set from memory

Syntax

```
reset_ssn [-quiet] [-verbose] <name>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<name>	Name of the set of results

Categories

[Report](#)

Description

Clear the SSN results from the specified named result set.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the set_msg_config command.

<name> - (Required) Specifies the name of the results to be cleared.

Examples

The following example clears the specified results set from memory:

```
reset_ssn SSN1
```

Send Feedback

See Also

- report_ssn

Send Feedback

reset_switching_activity

Reset switching activity on specified objects

Syntax

```
reset_switching_activity [-default] [-type <args>] [-hier] [-all]
    [-no_deassert_resets] [-quiet] [-verbose] [<objects>...]
```

Usage

Name	Description
[-default]	Reset default static probability and default toggle rate
[-type]	Specify nodes in a specific category. List of valid type values: io_output, io_bidir_enable, register, lut_ram, lut, dsp, bram_enable, bram_wr_enable, gt_txdata, gt_rxdata.
[-hier]	Hierarchically resets the switching activity on a hierarchical cells provided as <objects>.
[-all]	Reset switching activity on all nets
[-no_deassert_resets]	A switch to undo the deassertion of resets via command set_switching_activity -deassert_resets
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<objects>]	Objects to reset switching activity on

Categories

[Power](#), [XDC](#)

Description

Resets the attributes of the switching activity on specified nets, ports, pins, and cells in the design.

The switching activity is defined using the `set_switching_activity` command. The current switching activity defined for a specific port, pin, net, or cell can be found by using the `report_switching_activity` command.

Note: The `reset_switching_activity` is used to reset switching activity for specified objects. Use `set_switching_activity -default_toggle_rate` or `-default_static_probability` to change or reset the default values for the current design.

This command operates silently and does not return direct feedback of its operation.

Arguments

`-default` - (Optional) Reset the static probability and signal rate of the specified object.

Send Feedback

`-type <arg>` - (Optional) Reset the switching activity for the specified type of logic entity. By default, the command is applied to the top-level of the current design, or to the specified `<objects>`. The `-type` option applies the command to the specified type of logic objects in the top-level of the current design. The `-all` option, or `-hier` option, can be used to modify the scope of objects the command applies to. Valid logic types include:

- io_output - Primary outputs.

- io_bidir_enable - Enable pin of Bidir ports.

- register - All register outputs in the design/hierarchy specified.

- lut - All LUT outputs in the design/hierarchy specified.

- lut_ram - All distributed ram outputs in the design/hierarchy specified.

- dsp - All DSP outputs in the design/hierarchy specified.

- bram_enable - Enable pins (ENARDEN/ENBWREN) of BRAMs.

- bram_wr_enable - Write enables of BRAMs (WEA/WEBWE).

- gt_txdata - Output TX data pins of all GTs.

- gt_rxdata - Output RX data pins of all GTs.

`-hier` - (Optional) Reset the switching activity across all levels of the specified hierarchical object. Without `-hier`, the switching activity is applied to the specified `<objects>` at the current level of the hierarchy.

`-all` - (Optional) Must be used with `-type`, reset the switching activity on nets within all instances of the specified `-type` of logic object.

`-no_deassert_resets` - (Optional) Disables the `-deassert_resets` option if it was previously enabled using `set_switching_activity`.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<objects>` - (Optional) The list of objects for which to reset the switching activity. If not specified, the command resets the switching activity on all objects.

Examples

The following example resets the signal_rate and static probability value on all output ports:

```
reset_switching_activity -default [all_outputs]
```

See Also

- power_opt_design
- report_power
- report_switching_activity
- set_switching_activity

reset_target

Reset target data for the specified source

Syntax

```
reset_target [-quiet] [-verbose] <name> <objects>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<name>	List of targets to be reset, or 'all' to reset all generated targets
<objects>	The objects for which data needs to be reset

Categories

Project, IPFlow

Description

Remove the current target data for the specified IP core. This deletes any files that were delivered during generation of the specified targets. This does not remove the core from the current project, but does remove the associated target data from its referenced location.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the set_msg_config command.

<name> - (Required) Specifies the name of the type of target to reset. Valid values are:

- all - Reset all targets for the specified core.

- synthesis - Reset the synthesis netlist for the specified core. This will remove the netlist files for the specified core.

- `simulation` - Reset the simulation netlist for the specified core.

- `instantiation_template` - Reset the instantiation template for the specified core.

`<objects>` - (Required) The IP core objects to remove the target data from.

Examples

The following example resets the instantiation template for the specified IP core:

```
reset_target instantiation_template [get_ips blk_mem*]
```

See Also

- generate_target

Send Feedback

reset_timing

Resets the timing information on the current design

Syntax

```
reset_timing [-invalid] [-clock_reservation] [-quiet] [-verbose]
```

Usage

Name	Description
[-invalid]	Resets invalid timing constraints in addition to valid timing constraints.
[-clock_reservation]	Resets clock name reservations for auto-derived clocks in addition to valid timing constraints.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Report, Timing

Description

Reset the timing data and constraints for the current design. Use this command to clear the current in-memory timing data and constraints, and force the timing engine to reevaluate the design comprehensively rather than iteratively.

After clearing the constraints from the in-memory design, you must reload any needed constraints using the `read_xdc` command. The Vivado tool will not automatically reload the constraints.

> **TIP:** *This command deletes the in-memory timing view, not the timing report. Use the* `delete_timing_results` *command to delete the reported timing information.*

Arguments

`-invalid` - (Optional) Remove the invalid timing constraints as well as the valid timing constraints when resetting the design. Invalid constraints contain an error or are assigned to missing design objects, and are ignored by the Vivado timing engine at the time the XDC file is read, and so do not affect timing results. Resetting invalid constraints removes them from the in-memory design, so they will be lost if not previously saved to a constraints file.

`-clock_reservation` - (Optional) Resets auto-generated clock names as well as valid timing constraints. This allows the Vivado timing engine to regenerate the names of auto-generated clocks without regard to prior reserved names.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example clears the current timing data from memory, including any invalid timing constraints:

```
reset_timing -invalid
```

See Also

- delete_timing_results
- report_timing
- report_timing_summary

Send Feedback

resize_net_bus

Resize net bus in the current design

Syntax

```
resize_net_bus [-from <arg>] [-to <arg>] [-quiet] [-verbose]
    <net_bus_name>...
```

Usage

Name	Description
[-from]	New starting bus index
[-to]	New ending bus index
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<net_bus_name>	Name of the net bus to resize

Categories

Netlist

Description

Resize an existing bus net, to grow the bus, shrink the bus, or renumber the current range of indexes. You can only do a single grow, shrink, or renumber operation with each command.

- You can grow the bus by indicating a new range of indexes outside the current range of indexes. Growing the bus leaves existing bits connected as they currently are.

- You can shrink the bus by indicating a new range of indexes inside the current range of indexes. Shrinking the bus, eliminates connections to removed bits, but leaves the remaining bits connected as they currently are.

- You can renumber the current bus indexes by providing a new range of indexes with the same width as the current range. Renumbering bits changes bus bit numeric identifiers, but doesn't otherwise change connections.

Netlist editing changes the in-memory view of the netlist in the current design. It does not change the files in the source fileset, or change the persistent design on the disk. Changes made to the netlist may be saved to a design checkpoint using the `write_checkpoint` command, or may be exported to a netlist file such as Verilog, VHDL, or EDIF, using the appropriate `write_*` command.

Note: Netlist editing is not allowed on the elaborated RTL design.

This command returns nothing if successful, and returns an error if it fails.

Arguments

`-from <arg>` - (Optional) The new starting index of the specified bus net.

`-to <arg>` - (Optional) The new ending index of the specified bus.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<net_bus_name> - (Required) The names of an existing bus net.

Example

The following example creates a new 24-bit bus, then renumbers the bus indexes to include negative indexes, and then resizes the bus to shrink it to an 8-bit bus:

```
create_net tempBus -from 23 -to 0
resize_net_bus tempBus -from -12 -to 11
resize_net_bus tempBus -from 0 -to 7
```

See Also

- connect_net
- create_net
- create_pin
- create_port
- disconnect_net
- get_nets
- remove_net
- resize_pin_bus
- resize_port_bus
- write_checkpoint
- write_edif
- write_verilog
- write_vhdl

Send Feedback

resize_pblock

Move, resize, add and remove Pblock site-range constraints

Syntax

```
resize_pblock [-add <args>] [-remove <args>] [-from <args>] [-to <args>]
     [-replace] [-locs <arg>] [-quiet] [-verbose] <pblock>
```

Usage

Name	Description
[-add]	Add site ranges(s)
[-remove]	Remove site ranges(s)
[-from]	Site range(s) to move
[-to]	Site range destination(s)
[-replace]	Remove all existing ranges
[-locs]	LOC treatment Default: keep_all
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<pblock>	Pblock to resize

Categories

[Floorplan](#), [XDC](#)

Description

Place, resize, move, or remove the specified Pblock. The Pblock must have been created using the `create_pblock` command.

A Pblock consists of a group of cells that can be assigned to one or more independent or overlapping rectangles. Using the various options defined below, you can add sites to a rectangle, or remove sites from a rectangle, or define a new rectangle to be associated with an existing Pblock.

Arguments

`-add` <args> - (Optional) Add the specified range of sites to the Pblock. The range is specified as a rectangle from one corner to the diagonally opposite corner. The specific objects that can be used to define a Pblock range include the following: SLICE, DSP48, RAMB18, RAMB36, CLOCKREGION. For example SLICE_X0Y0:SLICE_X20Y12.

Note: Multiple site types are added as separate rectangles.

`-remove` <args> - (Optional) Remove the specified range of sites from the Pblock. Removing sites from a Pblock may result in the Pblock being broken into multiple smaller rectangles to enforce the requirement that Pblocks are defined as one or more rectangles.

`-from` <args> - (Optional) The `-from` and `-to` options must be used as a pair, and specify a site or range of sites to relocate from one location to another.

`-to` <args> - (Optional) The `-from` and `-to` options must be used as a pair, and specify a site or range of sites to relocate from one location to another.

`-locs` <args> - (Optional) Specifies how the placed logic in the Pblock will be handled as the Pblock is moved or resized. Valid values are:

- `keep_all` - leave all locs placed as they are currently. This is the default setting when `-locs` is not specified. Logic that is placed outside of the Pblock will no longer be assigned to the Pblock.

- `keep_only_fixed` - Specifies that only user-placed logic (fixed) will be preserved. Unfixed placed logic will be unplaced.

- `keep_none` - Unplace all logic.

- `move` - Specifies that all locs should be moved relative to the coordinates of the Pblock.

- `move_unfixed` - Specifies that only the unfixed placed elements should be moved. Logic placed by the user (fixed) will not be moved.

- `trim` - Specifies that logic that falls outside of the new Pblock boundaries will be unplaced. Any placed logic that still falls inside of the Pblock boundary will be left placed as it is.

- `trim_unfixed` - Trim only the unfixed placed logic.

`-replace` - (Optional) Remove all rectangles associated with the Pblock.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<pblock> - (Required) Specify the Pblock to be resized, moved, or removed.

Examples

The following example resizes the Pblock by adding a range of SLICEs, and removing other SLICEs, but keeps all instances placed at their current location:

```
resize_pblock block3 -add SLICE_X6Y67:SLICE_X11Y71 \
    -remove SLICE_X6Y71:SLICE_X7Y71 -locs keep_all
```

This example create a Pblock region, and defines the Pblock area by adding a range of CLOCKREGIONs:

```
create_pblock pblock_1
resize_pblock pblock_1 -add {CLOCKREGION_X0Y10:CLOCKREGION_X1Y11}
```

The following example moves the specified Pblock by adding a range of SLICEs, removing the existing range of SLICEs, and trims any placed logic that falls outside the new Pblock. Then it adds a new range of SLICEs and block ram to the specified Pblock in a second separate rectangle:

```
resize_pblock block3 -add SLICE_X3Y8:SLICE_X10Y3 \
    -remove SLICE_X6Y67:SLICE_X11Y71 -locs trim
resize_pblock block3 -add {SLICE_X6Y67:SLICE_X11Y71 \
    RAMB18_X0Y2:RAMB18_X1Y4}
```

See Also

- add_cells_to_pblock
- create_pblock

resize_pin_bus

Resize pin bus in the current design

Syntax

```
resize_pin_bus [-from <arg>] [-to <arg>] [-quiet] [-verbose]
    <pin_bus_name>...
```

Usage

Name	Description
[-from]	New starting bus index
[-to]	New ending bus index
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<pin_bus_name>	Name of the pin bus to resize

Categories

Netlist

Description

Resize an existing bus pin, to grow the bus, shrink the bus, or renumber the current range of pin indexes. You can only do a single grow, shrink, or renumber operation with each command.

- You can grow the bus by indicating a new range of pin indexes outside the current range of indexes. Growing the bus leaves existing pins connected as they currently are.

- You can shrink the bus by indicating a new range of pin indexes inside the current range of indexes. Shrinking the bus, eliminates connections to removed bus pins, but leaves the remaining pins connected as they currently are.

- You can renumber the current bus indexes by providing a new range of pin indexes with the same width as the current range. Renumbering pins changes the pin index, but does not otherwise change connections.

Netlist editing changes the in-memory view of the netlist in the current design. It does not change the files in the source fileset, or change the persistent design on the disk. Changes made to the netlist may be saved to a design checkpoint using the `write_checkpoint` command, or may be exported to a netlist file such as Verilog, VHDL, or EDIF, using the appropriate `write_*` command.

Note: Netlist editing is not allowed on the elaborated RTL design.

This command returns nothing if successful, and returns an error if it fails.

Arguments

`-from <arg>` - (Optional) The new starting index of the specified bus pin.

`-to <arg>` - (Optional) The new ending index of the specified bus pin.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<bus_pin_name> - (Required) The name of the bus pin to modify. You must specify the pin names hierarchically from the cell instance the pin is assigned to. Pins created at the top-level of the design are ports, and should be resized with the `resize_port_bus` command.

Examples

The following example creates a blackbox cell, then creates a 24-bit bidirectional bus for the specified hierarchical cell, then resizes the bus pin to expand the width to 32-bits, then renumbers the index to include negative bus indexes:

```
create_cell -reference dmaBlock -black_box usbEngine0/myDMA
create_pin -direction INOUT -from 0 -to 23 usbEngine0/myDMA/dataBus
resize_pin_bus -from 0 -to 31 usbEngine0/myDMA/dataBus
resize_pin_bus -from -16 -to 15 usbEngine0/myDMA/dataBus
```

See Also

- create_net
- create_pin
- create_port
- get_pins
- remove_pin
- resize_net_bus
- resize_port_bus
- write_checkpoint
- write_edif
- write_verilog
- write_vhdl

resize_port_bus

Resize port bus in the current design

Syntax

```
resize_port_bus [-from <arg>] [-to <arg>] [-quiet] [-verbose]
    <port_bus_name>...
```

Usage

Name	Description
[-from]	New starting bus index
[-to]	New ending bus index
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<port_bus_name>	Name of the port bus to resize

Categories

PinPlanning

Description

Resize an existing bus port, to grow the bus, shrink the bus, or renumber the current range of port indexes. You can only do a single grow, shrink, or renumber operation with each command.

- You can grow the bus by indicating a new range of port indexes outside the current range of indexes. Growing the bus leaves existing port indexes connected as they currently are.

- You can shrink the bus by indicating a new range of port indexes inside the current range of indexes. Shrinking the bus, eliminates connections to removed bus ports, but leaves the remaining ports connected as they currently are.

- You can renumber the current bus indexes by providing a new range of port indexes with the same width as the current range. Renumbering ports changes the port index, but does not otherwise change connections.

Netlist editing changes the in-memory view of the netlist in the current design. It does not change the files in the source fileset, or change the persistent design on the disk. Changes made to the netlist may be saved to a design checkpoint using the `write_checkpoint` command, or may be exported to a netlist file such as Verilog, VHDL, or EDIF, using the appropriate `write_*` command.

Note: Netlist editing is not allowed on the elaborated RTL design.

This command returns nothing if successful, and returns an error if it fails.

Arguments

`-from <arg>` - (Optional) The new starting index of the specified bus port.

`-to <arg>` - (Optional) The new ending index of the specified bus port.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<bus_port_name> - (Required) The name of the bus port to modify.

Examples

The following example creates a 32-bit output bus port, then renumbers the ports to include negative bus indexes, then shrinks the bus width from 32-bits to 16-bits:

```
create_port -direction out -from 0 -to 31 outPorts
resize_port_bus -from -16 -to 15 outPorts
resize_port_bus -from -8 -to 7 outPorts
```

See Also

- create_net
- create_pin
- create_port
- get_ports
- remove_port
- resize_net_bus
- resize_pin_bus
- write_checkpoint
- write_edif
- write_verilog
- write_vhdl

restart

Rewind simulation to post loading state (as if design was reloaded), time is set to 0

Syntax

```
restart [-quiet] [-verbose]
```

Returns

Restart retains breakpoints, Tcl forces, and settings in the waveform viewer but clears up the effects of all other Tcl commands.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Simulation

Description

Return the current simulation to its initial state, as if the design was reloaded, resetting the current simulation time to 0.

The `restart` command retains breakpoints, Tcl forces, and settings in the waveform configuration window, but resets all simulation values, and clears all other Tcl commands.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example restarts the current simulation:

```
restart
```

See Also

- current_time
- run
- stop

Send Feedback

resume_hw_hbm_amon

Resume Activity Monitor runs after paused for the specified hardware HBM(s)

Syntax

```
resume_hw_hbm_amon [-quiet] [-verbose] <hw_objects>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hw_objects>	hardware objects

Categories

Hardware

Description

The `resume_hw_hbm_amon` command restarts a running HBM activity monitor in the Vivado hardware manager that has been previously paused using the `pause_hw_hbm_amon` command.

This command returns nothing when successful, or returns an error when it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_objects> - (Required) Defines the HBM activity monitor objects (hw_hbms) to resume.

Examples

The following example restarts the HBM activity monitor for the associated HBM core:

```
resume_hw_hbm_amon [get_hw_hbms *HBM_2]
```

See Also

- add_hw_hbm_pc
- commit_hw_hbm
- current_hw_device
- current_hw_target
- get_hw_hbms
- pause_hw_hbm_amon
- refresh_hw_hbm
- remove_hw_hbm_pc
- run_hw_hbm_amon
- stop_hw_hbm_amon

route_design

Route the current design

Syntax

```
route_design [-unroute] [-release_memory] [-nets <args>] [-physical_nets]
    [-pins <arg>] [-directive <arg>] [-tns_cleanup] [-no_timing_driven]
    [-preserve] [-delay] [-auto_delay] -max_delay <arg> -min_delay <arg>
    [-timing_summary] [-finalize] [-ultrathreads] [-eco] [-quiet]
    [-verbose]
```

Usage

Name	Description
[-unroute]	Unroute whole design or the given nets/pins if used with -nets or -pins.
[-release_memory]	Release Router memory. Not compatible with any other options.
[-nets]	Operate on the given nets.
[-physical_nets]	Operate on all physical nets.
[-pins]	Operate on the given pins.
[-directive]	Mode of behavior (directive) for this command. Please refer to Arguments section of this help for values for this option. Default: Default
[-tns_cleanup]	Do optional TNS clean up.
[-no_timing_driven]	Do not run in timing driven mode.
[-preserve]	Preserve existing routing.
[-delay]	Use with -nets or -pins option to route in delay driven mode.
[-auto_delay]	Use with -nets or -pins option to route in constraint driven mode.
-max_delay	Use with -pins option to specify the max_delay constraint on the pins.When specified -delay is implicit.
-min_delay	Use with -pins option to specify the max_delay constraint on the pins.When specified -delay is implicit.
[-timing_summary]	Enable post-router signoff timing summary.
[-finalize]	finalize route_design in interactive mode.
[-ultrathreads]	Enable Turbo mode routing.
[-eco]	runs incremental router if there was eco modification on routed netlist.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Tools](#)

Send Feedback

Description

Route the nets in the current design to complete logic connections on the target part.

Predefined routing strategies can be quickly selected using the `route_design -directive` command, or specific route options can be configured to define your own routing strategy.

Routing can be completed automatically with `route_design`, or can be completed iteratively using the various options of the `route_design` command to achieve route completion and timing closure. Iterative routing provides you some control over the routing process to route critical nets first and then route less critical nets, and to control the level of effort and the timing algorithms for these various route passes.

Routing is one step of the complete design implementation process, which can be run automatically through the use of the `launch_runs` command when running the Vivado tools in Project Mode.

In Non-Project Mode, the implementation process must be run manually with the individual commands: `opt_design`, `place_design`, `phys_opt_design`, `power_opt_design`, and `route_design`. Refer to the *Vivado Design Suite User Guide: Design Flows Overview (UG892)* for a complete description of Project Mode and Non-Project Mode.

> 💡 **TIP:** *The* `route_design` *can be multi-threaded to speed the process. Refer to the* `set_param` *command for more information on setting the* `general.maxThreads` *parameter.*

Both placement and routing can be completed incrementally, based on prior results stored in a Design Checkpoint file (DCP), using the incremental implementation flow. Refer to the `read_checkpoint` command, or to *Vivado Design Suite User Guide: Implementation (UG904)* for more information on incremental place and route.

This command requires a placed design, and it is recommended that you have optimized the design with `opt_design` prior to placement.

Arguments

`-unroute <arg>` - (Optional) Unroute nets in the design. If no arguments are specified, all nets in the design are unrouted. The `route_design` command will not route any nets when the `-unroute` option is specified.

- Combine with the `-nets` option to limit unrouting to a list of nets.

- Combine with the `-pins` option to unroute from a specified pin to the nearest branch of the net.

- Combine with the `-physical_nets` option to unroute all logic 1 and logic 0 nets.

`-release_memory` - (Optional) Free router memory resources for subsequent route passes. This option does not run route passes, but only releases memory held by the router to reduce router initialization. The router will need to reload design data for subsequent route passes.

`-nets` <args> - (Optional) Route or unroute only the specified net objects. Net objects must be specified using the `get_nets` command.

> **TIP:** *The router uses a quick route approach to find a routing solution for the specified nets, ignoring timing delays, when routing with `-nets`, `-physical_nets`, or `-pins` specified. Use `-delay` to find a route with the shortest delay.*

`-physical_nets` - (Optional) Route or unroute only logic zero and logic one nets.

`-pins` <args> - (Optional) Route or unroute to the specified pins, which must be input pins. If a specified pin is driven by a multiple fanout net, only the route segment between the net and pin is affected.

`-directive <arg>` - (Optional) Direct routing to achieve specific design objectives. Only one directive can be specified for a single `route_design` command, and values are case-sensitive. Supported values are:

- `Explore` - Causes the Vivado router to explore different critical path routes based on timing, after an initial route.

- `AggressiveExplore` - Directs the router to further expand its exploration of critical path routes while maintaining original timing budgets. The router runtime might be significantly higher compared to the `Explore` directive as the router uses more aggressive optimization thresholds to attempt to meet timing constraints.

- `NoTimingRelaxation` - Prevents the router from relaxing timing to complete routing. If the router has difficulty meeting timing, it will run longer to try to meet the original timing constraints.

- `MoreGlobalIterations` - Uses detailed timing analysis throughout all stages instead of just the final stages, and will run more global iterations even when timing improves only slightly.

- `HigherDelayCost` - Adjusts the router's internal cost functions to emphasize delay over iterations, allowing a trade-off of runtime for better performance.

- `AdvancedSkewModeling` - Uses more accurate skew modeling throughout all routing stages which may improve design performance on higher-skew clock networks.

- `AlternateCLBRouting` - (UltraScale only) Chooses alternate routing algorithms that require extra runtime but may help resolve routing congestion.

- `RuntimeOptimized` - Run fewest iterations, trade higher design performance for faster runtime.

- `Quick` - Absolute fastest runtime, non-timing-driven, performs the minimum required routing for a legal design.

- `Default` - Run `route_design` with default settings.

> **IMPORTANT!** *The `-directive` option controls the overall routing strategy, and is not compatible with any specific `route_design` options, except `-preserve` and `-tns_cleanup`. It can also be used with `-quiet` and `-verbose`. Only the `Explore`, `Quick`, and `Default` directives are compatible with high reuse designs and the incremental implementation flow as defined by `read_checkpoint -incremental`. Refer to the Vivado Design Suite User Guide: Implementation (UG904) for more information on the use of the `-directive` option.*

`-tns_cleanup` - (Optional) By default, to reduce runtime, the router focuses on optimizing the Worst Negative Slack (WNS) path as opposed to Total Negative Slack (TNS) paths. This option invokes an optional phase at the end of routing where the router attempts to fix the TNS paths, those failing paths other than the WNS path. This option may reduce TNS at the cost of added runtime, but will not affect WNS. The `-tns_cleanup` option is recommended when using post-route `phys_opt_design` to ensure that optimization focuses on the WNS path and does not waste effort on TNS paths that can be fixed by the router. This option can be used in combination with `-directive`.

`-no_timing_driven` - (Optional) Disables the default timing driven routing algorithm. This results in faster routing results, but ignores any timing constraints during the routing process.

`-preserve` - (Optional) Existing completed routes will be preserved and not subject to the rip-up and reroute phase. This does not apply to routing that is fixed using the IS_ROUTE_FIXED or FIXED_ROUTE properties, which is not subject to being rerouted. Routing is preserved only for the current `route_design` command.

> **TIP:** *Partially routed nets are subject to rerouting to complete the connection. If you want to preserve the routing of a partially routed net, you should apply the FIXED_ROUTE property to the portion of the route you want to preserve.*

`-delay` - (Optional) Can only be used in combination with the `-nets` or `-pins` options. By default nets are routed to achieve the fastest routing runtime, ignoring timing constraints, when using `-nets` and `-pins` options. The `-delay` option directs the router to try to achieve the shortest routed interconnect delay, but still ignores timing constraints.

> **IMPORTANT!** *You can specify multiple nets to route at the same time using the `-delay` option, but this can result in conflicts for routing resources. The Vivado router may create node overlap errors if the nets are in close proximity to each other because the `-delay` option will reuse routing resources to achieve the shortest routes for all specified nets. Therefore it is recommended to route nets and pins individually using the -delay option, beginning with the most critical.*

`-auto_delay` - (Optional) Can only be used in combination with the `-nets` or `-pins` options. It is recommended to use the `-auto_delay` option on a placed design, and limit the specified number of nets or pins to less than 100. The `-auto_delay` option directs the router to prioritize setup and hold critical paths using the defined timing constraints.

`-max_delay <arg>` - (Optional) Can only be used with `-pins`. Directs the router to try to achieve an interconnect delay less than or equal to the specified delay given in picoseconds. When this options is specified, the `-delay` option is implied.

Note: The `-max_delay` and `-min_delay` options specify the delay limits for the interconnect only, not the logic or intra-site delay. The router attempts to satisfy the delay restrictions on the interconnect.

`-min_delay <arg>` - (Optional) Can only be used with `-pins`. Directs the router to try to achieve an interconnect delay greater than or equal to the specified delay given in picoseconds. When this option is specified, the `-delay` option is implied.

`-timing_summary` - (Optional) By default, the router outputs a final timing summary to the log, based on Vivado router internal estimated timing which might differ slightly from the actual routed timing due to pessimism in the delay estimates. The `-timing_summary` option forces the router to launch the Vivado static timing analyzer to report the timing summary based on actual routed delays, but incurs additional run time for the static timing analysis. The timing summary consists of the Worst Negative Slack (WNS), Total Negative Slack (TNS), Worst Hold Slack (WHS), and Total Hold Slack (THS). The values are identical to that of `report_timing_summary` when run on the post-route design.

> **TIP:** *The Vivado static timing analyzer is also launched by the* `-directive Explore` *option.*

`-finalize` - (Optional) When routing interactively you can specify `route_design -finalize` to complete any partially routed connections.

`-ultrathreads` - (Optional) Reduces router runtime at the expense of repeatability. This option enables the router to run faster, but there will be some variation in routing results between otherwise identical runs.

`-eco` - (Optional) Run router in incremental mode when there is an eco modification. When in incremental mode, as well as routing new nets, the router can modify existing nets to improve routability and the ability to meet timing. This option can be called by itself, or it can be used with the `-no_timing_driven` switch or the `Quick` directive. These options run the router with timing off for a faster result.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

Route the entire design, and direct the router to try multiple algorithms for improving critical path delay:

```
route_design -directive Explore
```

The following example routes the set of timing critical nets, $criticalNets, to the shortest interconnect delay, marks the nets as fixed using the IS_ROUTE_FIXED property, and then routes the rest of the design using a low effort directive for fast results:

```
route_design -delay -nets $criticalNets
set_property IS_ROUTE_FIXED 1 $criticalNets
route_design -directive RuntimeOptimized
```

Route the specified nets using the fastest runtime:

```
route_design -nets [get_nets ctrl0/ctr*]
```

Route the specified nets to get the shortest interconnect delays:

```
route_design -nets [get_nets ctrl0/ctr*] -delay
```

Route to the specified pins:

```
route_design -pins [get_pins ctrl0/reset_reg/D ctrl0/ram0/ADDRARDADDR]
```

Route to a particular pin, try to achieve less than 500 ps delay:

```
route_design -pins [get_pins ctrl0/reset_reg/D] -max_delay 500
```

Route to a particular pin, try to achieve more than 200 ps delay:

```
route_design -pins [get_pins ctrl0/ram0/ADDRARDADDR] -min_delay 200
```

See Also

- get_nets
- get_pins
- launch_runs
- opt_design
- phys_opt_design
- place_design
- power_opt_design
- read_checkpoint
- set_property
- write_checkpoint

Send Feedback

run

Run the simulation for the specified time

Syntax

```
run [-all] [-quiet] [-verbose] [<time>] [<unit>]
```

Usage

Name	Description
[-all]	Runs simulation till a breakpoint, an exception or no events left in the queue
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<time>]	Length of simulation time
[<unit>]	Unit for time from the following time units: fs, ps, ns, us, ms, sec

Categories

Simulation

Description

Run the current simulation from the current time to the specified time, or until the simulation stops.

A running simulation can be stopped at a predetermined time, at a specific breakpoint in the HDL source code, by encountering a TRUE condition, by evaluating the circuit until there are no remaining events, or by encountering a runtime error such as an out-of-bounds value.

The `run` command instructs an existing simulation to run for a specified length of time, or until there are no remaining events. The time is specified as a floating point number indicating a period of time in the current simulation units, or in the specified units.

Arguments

`-all` - Run the simulation until no event is left in the event queue, a breakpoint or valid condition is encountered, or a run time exception occurs.

Note: This is the default when no other options are specified.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Send Feedback

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<time> - (Optional) Specifies the length of the simulation run time. The time can be specified as a floating point number, or using the keyword `all`.

Note: Using the keyword `all` is the same as using the `-all` option.

<unit> - (Optional) One of the following time units (with or without space between time and unit) can be specified: `fs`, `ps`, `ns`, `us`, `ms`, or `sec`. The default is the value of `time_unit`.

Examples

The following example runs an existing simulation for the specified simulation run time, using the default units (ns):

```
run 1000
```

The following example runs an existing simulation for 300 microseconds (us):

```
run 300 us
```

The following example runs the current simulation until no event is left in the event queue, a breakpoint or valid condition is met, or a simulation runtime error occurs:

```
run -all
```

See Also

- add_bp
- add_condition
- restart
- step
- stop

run_hw_axi

Run hardware AXI read/write transaction(s)and update transaction status in hw_axi object..

Syntax

```
run_hw_axi [-queue] [-quiet] [-verbose] <hw_axi_txns>...
```

Usage

Name	Description
[-queue]	Queue Transaction. Default: 0
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hw_axi_txns>	hardware AXI Transaction object to execute on the AXI bus.

Categories

Hardware

Description

Run the AXI transactions defined on the specified JTAG to AXI Master core.

AXI transactions are created with the create_hw_axi_txns command.

Run the specified hardware AXI read/write transactions on the AXI bus, and update the transaction status on the associated hw_axi object.

Arguments

-queue - (Optional) Run the specified hw_axi transactions in queue mode. Queued operation allows up to 16 read and 16 write transactions to be queued in the JTAG to AXI Master FIFO and issued back-to-back for low latency and higher performance between the transactions. Non-queued transactions are simply run as submitted.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_axi_txns> - (Required) Specify the hardware AXI Transaction objects to run on the AXI bus. The objects can be returned by the `get_hw_axi_txns` command.

Example

The following example runs the AXI transactions currently defined on the specified hw_axi object:

```
run_hw_axi [get_hw_axi_txns [get_hw_axis]]
```

This example runs four AXI transactions in queued mode:

```
run_hw_axi -queue  [get_hw_axi_txns txn_1] [get_hw_axi_txns txn_2] \
[get_hw_axi_txns txn_3] [get_hw_axi_txns txn_4]
```

This example creates AXI read and write transactions, runs the hw_axi, and reports on the results:

```
create_hw_axi_txn wr_txn [lindex [get_hw_axis] 0] -address 80000000 \
-data {11112222 33334444 55556666 77778888} -len 4 -type write
create_hw_axi_txn rd_txn [lindex [get_hw_axis] 0] -address 80000000 \
-len 4 -type read
run_hw_axi [get_hw_axi_txns wr_txn]
set wr_report [report_hw_axi_txn wr_txn -w 32]
puts $wr_report
run_hw_axi [get_hw_axi_txns rd_txn]
set rd_report [report_hw_axi_txn rd_txn -w 32]
puts $rd_report
close_hw_target;
disconnect_hw_server;
```

See Also

- create_hw_axi_txn
- delete_hw_axi_txn
- get_hw_axis
- get_hw_axi_txns
- refresh_hw_axi
- reset_hw_axi

Send Feedback

run_hw_hbm_amon

Enable Activity Monitor runs for the specified hardware HBM(s)

Syntax

```
run_hw_hbm_amon [-quiet] [-verbose] <hw_objects>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hw_objects>	hardware objects

Categories

Hardware

Description

The HBM activity monitor (amon) can be used to gain real-time access to performance monitoring and temperature sensors certain Xilinx® UltraScale+ devices that include an integrated High-Bandwidth Memory (HBM) controller. Refer to the LogiCore IP Product Guide: AXI High Bandwidth Memory Controller ([PG276](PG276)) for more information on this core. The HBM controller and memory stacks contain both performance counters and temperature sensors that can be accessed through the HBM activity monitor from within the Xilinx Vivado hardware manager. The HBM activity monitor displays read, write, and overall throughput of data, as well as the device temperature. The activity monitor displays, captures, and exports data to a CSV file when it is running.

The `run_hw_hbm_amon` command runs an HBM activity monitor in the Vivado hardware manager that has been previously configured using the `add_hw_hbm_pc` command.

This command returns the file path and name of a CSV file the collected data will be written to, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_objects> - (Required) Specifies the specific HBM activity monitor objects (hw_hbms) to run.

Examples

The following example runs the HBM activity monitor for the associated HBM core:

```
run_hw_hbm_amon [get_hw_hbms *HBM_2]
```

See Also

- add_hw_hbm_pc
- commit_hw_hbm
- current_hw_device
- current_hw_target
- get_hw_hbms
- pause_hw_hbm_amon
- refresh_hw_hbm
- remove_hw_hbm_pc
- resume_hw_hbm_amon
- stop_hw_hbm_amon

Send Feedback

run_hw_ila

Arm hardware ILAs.

Syntax

```
run_hw_ila [-trigger_now] [-compile_only] [-file <arg>] [-force] [-quiet]
    [-verbose] [<hw_ilas>...]
```

Usage

Name	Description
[-trigger_now]	Trigger and capture immediately.
[-compile_only]	Test only compile trigger state machine file but do not upload.
[-file]	Trigger at startup file name. Command will not arm ILA core.
[-force]	Overwrite existing file
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<hw_ilas>]	hardware ILAs Default: Current hardware ILA

Categories

Hardware

Description

Arm triggers and run the specified hardware ILA debug cores on the current hardware device.

The Integrated Logic Analyzer (ILA) debug core lets you perform in-system debug of implemented designs, or design bitstreams, on a programmed Xilinx FPGA. The ILA core includes many advanced features of modern logic analyzers, including boolean trigger equations, and edge transition triggers. You can use the ILA core to probe specific signals of the design, to trigger on programmed hardware events, and capture data samples from the Xilinx FPGA in real-time. Refer to *LogiCORE IP Integrated Logic Analyzer (PG172)* for details of the ILA core.

You can add ILA debug cores into the RTL source files of a design, or in the synthesized netlist using the `create_debug_core` command. Refer to the *Vivado Design Suite User Guide: Vivado Programming and Debugging (UG908)* for more information on adding debug cores and signal probes to the design.

Send Feedback

Debug cores and probes are written to a probes file (`.ltx`) using the `write_debug_probes` command, and associated with the hardware device along with the bitstream file (`.bit`) using the PROBES.FILE and PROGRAM.FILE properties of the hw_device object. The hardware device is programmed using the `program_hw_devices` command. The ILA debug cores in the design are accessible from the Hardware Manager using the `get_hw_ilas` command. The debug probes assigned to the ILA debug cores can be returned with the `get_hw_probes` command.

The steps to debug your design in hardware using an ILA debug core are:

1. Connect to the hardware server and target using `connect_hw_server` and `open_hw_target`.

2. Program the FPGA with the bitstream (`.bit`) and probes (`.ltx`) files using `program_hw_devices`.

3. Set up the ILA debug core trigger events and data capture controls using `set_property` to configure properties of the ILA.

4. Arm the ILA debug core triggers on the Xilinx FPGA using `run_hw_ila`. When a trigger event occurs, or the capture condition is met, the ILA capture buffer is filled.

5. Uploaded sampled data from the hw_device into a Vivado logic analyzer hw_ila_data object using `upload_hw_ila_data`.

6. View the captured data in the Waveform window of the Vivado logic analyzer feature using `display_hw_ila_data`.

You can set up an ILA debug core to trigger on specific events or conditions at the debug probes, and to capture data under specific conditions, using CONTROL properties on the hw_ila object. You set these properties with the `set_property` command. Refer to the *Vivado Design Suite User Guide: Vivado Programming and Debugging (UG908)* for more information on setting properties to configure debug cores and signal probes to monitor the design.

Note: The Vivado IDE provides a graphical interface to configure hw_ila and hw_probes for trigger and capture. You can use the Vivado IDE to see the properties needed to configure and run the hw_ila.

The specific properties on the hw_ila you can use to configure a debug core include the following:

- CONTROL.DATA_DEPTH - Defaults to the MAX_DATA_DEPTH of the ILA debug core, which was set when the debug core was created or inserted into the design. The data depth defines the number of data samples the hw_ila object can capture in a data window. Set the data depth as an integer which is a power of two, from 1 to the maximum data depth (MAX_DATA_DEPTH) of the hw_ila.

  Note: The value of DATA_DEPTH is related to CONTROL.WINDOW_COUNT by the equation:
  `DATA_DEPTH * WINDOW_COUNT = MAX_DATA_DEPTH`

Send Feedback

- CONTROL.WINDOW_COUNT - Lets you divide the MAX_DATA_DEPTH of the ILA core into a number of data windows to store sample data from multiple trigger events. In the case of 10 data windows for example, the first window will be filled at the first trigger event, and each subsequent window will be filled upon subsequent triggering events or capture conditions.

- CONTROL.TRIGGER_POSITION - An integer value related to the DATA_DEPTH. Positions the trigger event in the sample data buffer. For a DATA_DEPTH of 1024, position 0 refers to the first (or left-most) data buffer and 1023 refers to the last (or right-most) data buffer. The TRIGGER_POSITION lets you capture sample data ahead of the trigger event. For instance, with a DATA_DEPTH of 256, a TRIGGER_POSITION of 100 allows you to capture 100 data samples ahead of the trigger, and 155 data samples at and after the trigger event.

  Note: In the case of `run_hw_ila -trigger_now 1`, the TRIGGER_POSITION merely positions the trigger mark in the Vivado logic analyzer waveform window. Because the trigger event is immediate, there is no time to capture data samples ahead of the trigger.

- CONTROL.TRIGGER_MODE - Valid values include:

  ○ BASIC_ONLY - The trigger condition is the result of a Boolean equation using the TRIGGER_CONDITION to evaluate the values on each of the associated ILA probes.

  ○ BASIC_OR_TRIG_IN - The ILA core is triggered by a Boolean equation considering probe values, or by the TRIG_IN port on the core.

  ○ ADVANCED_ONLY - The ILA core is configured to have advanced trigger capabilities defined in a user-defined Trigger State Machine (TSM).

  ○ ADVANCED_OR_TRIG_IN - The ILA core is triggered by the TSM or by the TRIG_IN port on the core.

  ○ TRIG_IN_ONLY - The ILA core is triggered only by the TRIG_IN port on the core.

- CONTROL.TRIGGER_CONDITION - Defines a Boolean equation which evaluates comparators on participating probes on the ILA debug core. When the condition evaluates to true, the BASIC trigger mode is satisfied. Valid values include:

  ○ AND - Trigger condition is "true" if all participating probe comparators evaluate "true", otherwise trigger condition is "false".

  ○ NAND - Trigger condition is "true" if at least one participating probe comparator evaluates "false", otherwise trigger condition is "false".

  ○ OR - Trigger condition is "true" if at least one participating probe comparator evaluates "true", otherwise trigger condition is "false".

  ○ NOR - Trigger condition is "true" if all participating probe comparators evaluate "false", otherwise trigger condition is "false".

  Note: The evaluation of the probes participating in the trigger condition is determined by the TRIGGER_COMPARE_VALUE property assigned to the hw_probe object, as returned by `get_hw_probes`. If the TRIGGER_COMPARE_VALUE is 'X' then it is not participating in the trigger condition.

- CONTROL.TSM_FILE - Specify the path to a file defining a Trigger Finite State Machine (TSM) to be used for advanced trigger handling.

- CONTROL.TRIG_OUT_MODE - Used to transition the TRIG_OUT port on the ILA core to be used to drive the TRIG_IN port on other ILA cores. Valid values include:

  - DISABLED - Disable the TRIG_OUT port on the ILA core.

  - TRIGGER_ONLY - Transition the TRIG_OUT port when the trigger conditions have been satisfied.

  - TRIG_IN_ONLY - Transition the TRIG_OUT when the TRIG_IN signal transitions. Use this to pass the trigger event along a chain of ILA cores.

  - TRIGGER_OR_TRIG_IN - Transition the TRIG_OUT when either the trigger conditions are satisfied, or the TRIG_IN transitions.

- CONTROL.CAPTURE_MODE - Valid values include ALWAYS or BASIC. Capture and store a data sample on the debug core ALWAYS during a given clock cycle, or only if the CAPTURE_CONDITION evaluates to "true" (BASIC).

- CONTROL.CAPTURE_CONDITION - Defines a Boolean equation for participating probe comparators on the ILA debug core that must evaluate to TRUE to meet the data capture condition. When the capture condition evaluates to true, the BASIC capture mode is satisfied. Valid values include:

  - AND - Capture condition is "true" if all participating probe comparators evaluate "true", otherwise capture condition is "false".

  - NAND - Capture condition is "true" if at least one participating probe comparator evaluates "false", otherwise capture condition is "false".

  - OR - Capture condition is "true" if at least one participating probe comparator evaluates "true", otherwise capture condition is "false".

  - NOR - Capture condition is "true" if all participating probe comparators evaluate "false", otherwise capture condition is "false".

Note: The evaluation of the probes participating in the capture condition is determined by the CAPTURE_COMPARE_VALUE property assigned to the hw_probe object, as returned by `get_hw_probes`. If the CAPTURE_COMPARE_VALUE is 'X' then it is not participating in the trigger condition.

> **TIP:** *There are other properties on the ILA core that also determine the operation of the core, but they are not user-configurable.*

With the ILA core configured, you can use the `run_hw_ila` command to arm the ILA cores on the target part. When this command is run, the trigger configurations defined in the hw_ila and hw_probe objects are written to the target Xilinx FPGA (hw_device) and arms the ILA core or cores on the device.

With the hw_ila armed and running, the `wait_on_hw_ila` command stops your Tcl script to wait for the data sample buffers to be populated with captured data. When the memory of the ILA core is full on the physical hw_device, the `wait_on_hw_ila` command returns, and your Tcl script resumes.

You can use `upload_hw_ila_data` to upload the captured data from the physical memory of the hw_device into a hw_ila_data object in the Vivado logic analyzer. Then view the ILA data in the waveform window of the Vivado logic analyzer using `display_hw_ila_data`, and write the data for use in external tools using the `write_hw_ila_data` command.

You can also immediately trigger the probes on the hw_device using the `-trigger_now` option, to capture data from the device right away, rather than waiting for trigger events or capture conditions to be met over time.

You can use `reset_hw_ila` to restore the CONTROL properties of the ILA debug core to their default setting, and reset the probe comparator values to 'X'.

Arguments

`-trigger_now` - (Optional) Immediately trigger the ILA debug cores, regardless of trigger conditions, to capture sample data at the debug ports. This is a boolean argument enabled by its presence.

`-compile_only` - (Optional) Compile the trigger state machine file (CONTROL.TSM_FILE) to perform syntax checking and report compiler errors and warnings. This option does not load and arm the hardware device. This option is only supported when in advanced trigger mode (CONTROL.TRIGGER_MODE ADVANCED_ONLY), and will otherwise return an error.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`-file <arg>` - (Optional) When the TRIGGER_MODE is set to BASIC_ONLY, this option will export a file containing the current trigger and probe configuration settings. The trigger file can be used by the `apply_hw_ila_trigger` command to define triggers for the hw_device to use on startup, or upon booting the device. Using the `-file` option prevents the `run_hw_ila` command from arming the ILA core triggers on the hardware device.

Note: If no extension is specified, a default file extension of `.tas` is assigned.

`-force` - (Optional) Overwrite the specified trigger file if it already exists. By default the tool will not overwrite an existing file.

<hw_ilas> - (Optional) Specify one or more hw_ila objects to run. The hw_ila objects can either be specified as objects returned by the `get_hw_ilas` or `current_hw_ila` commands, or specified by name. If the hw_ila is not specified, the `current_hw_ila` will be run.

> **IMPORTANT!** *You can only specify one hw_ila object when using the `-file` option.*

Example

The following example sets the trigger and capture properties on the current hw_ila object, and then runs the hw_ila, specified by name, to arm the hw_device and wait for the data buffers to be filled:

```
current_hw_ila [get_hw_ilas hw_ila_1]
set_property CONTROL.DATA_DEPTH 1024 [current_hw_ila]
set_property CONTROL.TRIGGER_MODE BASIC_ONLY [current_hw_ila]
set_property CONTROL.TRIGGER_CONDITION AND [current_hw_ila]
set_property CONTROL.CAPTURE_MODE ALWAYS [current_hw_ila]
run_hw_ila -trigger_now hw_ila_1
wait_on_hw_ila hw_ila_1
```

This example writes a trigger configuration file for the current hw_ila object, to the specified file, but does not arm the current hw_device:

```
run_hw_ila -file C:/Data/trigger_config1
```

Note: A default file extension of `.tas` is assigned to the specified file.

See Also

- apply_hw_ila_trigger
- current_hw_device
- current_hw_ila
- current_hw_ila_data
- get_hw_devices
- get_hw_ilas
- get_hw_ila_datas
- get_hw_probes
- reset_hw_ila
- set_property
- upload_hw_ila_data
- wait_on_hw_ila
- write_debug_probes
- write_hw_ila_data

run_hw_sio_scan

Run hardware SIO scans.

Syntax

```
run_hw_sio_scan [-quiet] [-verbose] <hw_sio_scans>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hw_sio_scans>	hardware SIO scans

Categories

[Hardware](#)

Description

Run the specified serial I/O analyzer link scan.

To analyze the margin of a given link, it is often helpful to run a scan of the link using the specialized Eye Scan hardware of Xilinx UltraScale devices or 7 series FPGAs. The Vivado serial I/O analyzer feature lets you to create, run, and save link scans.

This command creates and returns a link scan object that you can use with the `run_hw_sio_scan` command to run analysis on the specified links, or GT receivers. You can also save the scan to disk using the `write_hw_sio_scan` command.

This command run analysis on the specified scan objects. If running in a Tcl script, you can suspend the script while the scan completes using the `wait_on_hw_sio_scan` command. You can stop a running scan using the `stop_hw_sio_scan` command.

You can save the scan to disk using the `write_hw_sio_scan` command.

You can remove the created scan object using `remove_hw_sio_scan`.

This command returns the hw_sio_scan object, or returns an error if the command fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_sio_scans> - (Required) Specify one or more hw_sio_scan objects to run. The hw_sio_scans must be specified as objects as returned by the `create_hw_sio_scan` or `get_hw_sio_scans` commands.

Example

The following example runs the specified serial I/O analyzer scan:

```
run_hw_sio_scan [lindex [get_hw_sio_scans {SCAN_3}] 0]
```

See Also

- create_hw_sio_scan
- current_hw_device
- get_hw_sio_scans
- remove_hw_sio_scan
- stop_hw_sio_scan
- wait_on_hw_sio_scan
- write_hw_sio_scan

run_hw_sio_sweep

Run hardware SIO sweeps.

Syntax

```
run_hw_sio_sweep [-quiet] [-verbose] <hw_sio_sweeps>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hw_sio_sweeps>	hardware SIO sweeps

Categories

Hardware

Description

Run a serial I/O analyzer link sweep scan to run multiple scans across a range of values.

To analyze the margin of a given link, it is often helpful to run a scan of the link using the specialized features of Xilinx UltraScale devices or 7 series FPGAs. It can also be helpful to run multiple scans on a the link with different configuration settings for the GTs. This can help you determine which settings are best for your design. The Vivado serial I/O analyzer feature enables you to define, run, and save link sweeps, or collections of link scans run across a range of values.

This command run analysis on the specified sweep scan objects. If running in a Tcl script, you can suspend the script while the sweep scan completes using the `wait_on_hw_sio_sweep` command. You can stop a running sweep scan using the `stop_hw_sio_sweep` command.

You can save the sweep scan to disk using the `write_hw_sio_sweep` command.

You can remove the created scan object using `remove_hw_sio_sweep`.

This command returns the hw_sio_sweep object, or returns an error if the command fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_sio_sweeps> - (Required) Specify one or more hw_sio_sweep objects to run. The hw_sio_sweep must be specified as an object as returned by the `get_hw_sio_sweeps` command.

Example

The following example runs the specified sweep scan:

```
run_hw_sio_sweep [lindex [get_hw_sio_sweeps {SWEEP_0}] 0]
```

See Also

- create_hw_sio_sweep
- current_hw_device
- get_hw_sio_sweeps
- remove_hw_sio_sweep
- stop_hw_sio_sweep
- wait_on_hw_sio_sweep
- write_hw_sio_sweep

run_state_hw_jtag

change to a stable state of a specified transition

Syntax

```
run_state_hw_jtag [-state <args>] [-quiet] [-verbose] <stable_state>
```

Returns

Hardware JTAG

Usage

Name	Description
`[-state]`	valid state path sequence to stable_state
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<stable_state>`	valid stable_state - valid stable states IDLE, RESET, IRPAUSE, and DRPAUSE

Categories

Hardware, Object

Description

Transition the hw_jtag object of the current hardware target to the specified TAP stable state.

A hw_jtag object is created by the Hardware Manager feature of the Vivado Design Suite when a hardware target is opened in JTAG mode using the `open_hw_target -jtag_mode` command.

The `run_state_hw_jtag` command specifies:

- An ending or target TAP stable state to transition to.

- An optional state path list to transition through to get from the current state to the target state.

Send Feedback

If an optional `-state` path list is defined, then the state list must contain all states needed to reach the stable state, or the command will return an error. If no state path list is defined, then the command will transition from the current state to the target state according to the state transition paths defined in the following table:

```
Current     Target      State Transition Path
State       State
DRPAUSE     RESET       DRPAUSE -> DREXIT2 -> DRUPDATE -> DRSELECT ->
                            IRSELECT-> RESET
DRPAUSE     IDLE        DRPAUSE -> DREXIT2 -> DRUPDATE -> IDLE
DRPAUSE     DRPAUSE     DRPAUSE -> DREXIT2 -> DRUPDATE -> DRSELECT ->
                            DRCAPTURE -> DREXIT1 -> DRPAUSE
DRPAUSE     IRPAUSE     DRPAUSE -> DREXIT2 -> DRUPDATE -> DRSELECT ->
                            IRSELECT -> IRCAPTURE -> IREXIT12 -> IRPAUSE
IDLE        RESET       IDLE -> DRSELECT -> IRSELECT -> RESET
IDLE        IDLE        IDLE
IDLE        DRPAUSE     IDLE -> DRSELECT -> DRCAPTURE -> DREXIT1 -> DRPAUSE
IDLE        IRPAUSE     IDLE -> DRPAUSE -> IRSELECT ->IRCAPTURE ->
                            IREXIT1 -> IRPAUSE
IRPAUSE     RESET       IRPAUSE -> IREXIT2 -> IRUPDATE -> DRSELECT ->
                            IRSELECT -> RESET
IRPAUSE     IDLE        IRPAUSE -> IREXIT2 -> IRUPDATE -> IDLE
IRPAUSE     DRPAUSE     IRPAUSE -> IREXIT2 -> IRUPDATE -> DRSELECT ->
                            DRCAPTURE -> DREXIT1 -> DRPAUSE
IRPAUSE     IRPAUSE     IRPAUSE -> IREXIT2 -> IRUPDATE -> DRSELECT ->
                            IRSELECT -> IRCAPTURE -> IREXIT1 -> IRPAUSE
RESET       RESET       RESET
RESET       IDLE        RESET -> IDLE
RESET       DRPAUSE     RESET -> IDLE -> DRSELECT -> DRCAPTURE ->
                            DREXIT1 -> DRPAUSE
RESET       IRPAUSE     RESET -> IDLE -> DRSELECT -> IRSELECT ->
                            IRCAPTURE -> IREXIT1 -> IRPAUSE
```

This command returns the target stable state when successful, or returns an error if it fails.

Arguments

`-state` <args> - (Optional) A valid path sequence of states to transition the hw_jtag object from the current state to the target <stable_state>. Valid states include:

- IDLE

- RESET

- DRSELECT, DRCAPTURE, DRSHIFT, DRPAUSE, DREXIT1, DREXIT2, DRUPDATE

- IRSELECT, IRCAPTURE, IRSHIFT, IRPAUSE, IREXIT1, IREXIT2, IRUPDATE

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

Send Feedback

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the set_msg_config command.

<stable_state> - (Required) Valid stable target state, or end state. Valid target states include:

- IDLE

- RESET

- DRPAUSE

- IRPAUSE

Example

The following example transitions through various TAP stable states:

```
// Go to state RESET
run_state_hw_jtag RESET
// From current state RESET, go to DRPAUSE
run_state_hw_jtag DRPAUSE
// From DRPAUSE, go to IDLE state transitioning through
// the specified states
run_state_hw_jtag -state {DREXIT2 DRUPDATE IDLE} IDLE
// From IDLE, go to RESET, through the specified states
// note that specified path starts with an extra TCK
// clock cycle in the IDLE state
run_state_hw_jtag RESET -state {IDLE DRSELECT IRSELECT RESET}
```

See Also

- connect_hw_server
- current_hw_target
- open_hw_target
- runtest_hw_jtag
- scan_dr_hw_jtag
- scan_ir_hw_jtag

runtest_hw_jtag

Forces IEEE 1149.1 TAP state machine to a stable state for a specified wait period

Syntax

```
runtest_hw_jtag [-wait_state <arg>] [-end_state <arg>] [-sec <arg>]
    [-max_wait <arg>] [-tck <arg>] [-quiet] [-verbose]
```

Usage

Name	Description
[-wait_state]	valid stable_state - valid stable states IDLE, RESET, IRPAUSE, and DRPAUSE
[-end_state]	valid stable_state - valid stable states IDLE, RESET, IRPAUSE, and DRPAUSE
[-sec]	Number of seconds to wait in wait_state
[-max_wait]	Maximum Number of seconds to wait in wait_state - max timeout
[-tck]	Number of TCK cycles to wait in wait_state Default: Number of TCK cycles to wait in wait_state
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Hardware](), [Object]()

Description

Specify a wait operation for the hw_jtag object state machine which defines:

- Which TAP stable state to go to perform the wait operation.

- A wait time expressed as:

  - 'n' TCK cycles, where 'n' is a 32-bit unsigned decimal number.

  - A minimum and optionally maximum time in seconds to stay in the wait state, with min/max times specified as unsigned integers or real numbers.

- The TAP stable state to go after the wait operation has completed.

The default values for `-wait_state` and `-end_state` are IDLE. If a non-IDLE wait_state or end_state are defined, then the hw_jtag object will first transition to the specified wait_state before starting the wait operation. Once the wait time has elapsed, the hw_jtag object transitions to the specified end_state. When the wait_state and/or end_state are specified by the `runtest_hw_jtag` command, subsequent commands will use the same wait_state/end_state unless they are changed.

This command returns the end stable state, or returns an error if it fails.

Note: If the command cannot meet the wait time specification, then it will raise an exception that can be trapped by the Tcl `catch` command.

Arguments

`-wait_state <arg>` - (optional) Specify the state to to go to while in the wait state. Can be specified as one of the following TAP stable states: IDLE, RESET, IRPAUSE, or DRPAUSE. The default is IDLE.

`-end_state <arg>` - (optional) Specify the state to transition into after the wait operation has completed. Can be specified as one of the following TAP stable states: IDLE, RESET, IRPAUSE, or DRPAUSE. The default is IDLE.

`-sec <arg>` - (Optional) 32-bit decimal integer specifying the minimum number of seconds to wait.

`-max_wait <arg>` - (Optional) Maximum number of seconds to wait in wait_state.

`-tck <arg>` - (Optional) 32-bit decimal integer specifying the number of JTAG clock cycles to wait.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Example

The following example walks through a series of `runtest_hw_jtag` commands with various wait_states and end_states specified:

```
// Wait in default IDLE state for 1000 TCKs,
// then go to end_state DRPAUSE
runtest_hw_jtag -tck 1000 -end_state DRPAUSE
// Go from DRPAUSE (end_state defined in previous
// runtest_hw_jtag command) to IDLE and wait for
// 500 TCK clock cycles before going to DRPAUSE again
runtest_hw_jtag -tck 500
// Go from DRPAUSE to IDLE and wait for
 //1,000,000 TCKs or at least
// 5 seconds before transitioning to DRPAUSE
runtest_hw_jtag -tck 1000000 -sec 5
// Go from DRPAUSE to IDLE and wait for
// at least 1 millisecond and at most 50 milliseconds
// before remaining in IDLE state
runtest_hw_jtag -sec 1.0E-3 -max_wait 50.0E-3 -end_state IDLE
// Go from IDLE to DRPAUSE and wait for at least
// 85 milliseconds before returning to IDLE state
runtest_hw_jtag -wait_state DRPAUSE -sec 85E-3
// Go from IDLE to DRPAUSE state and wait for
// at least 1 second before returning to IDLE state
runtest_hw_jtag -sec 1
// Go to wait_state IDLE (note: current end_state is IDLE),
// wait for at least 10 milliseconds, then stay in IDLE state.
runtest_hw_jtag -wait_state IDLE -sec 1E-2
```

> **TIP:** *The wait_state or end_state from the first* `runtest_hw_jtag` *command will be used in subsequent commands unless specifically changed.*

See Also

- [connect_hw_server](#)
- [current_hw_target](#)
- [open_hw_target](#)
- [run_state_hw_jtag](#)
- [scan_dr_hw_jtag](#)
- [scan_ir_hw_jtag](#)

save_bd_design

Save an existing IP subsystem design to disk file.

Syntax

```
save_bd_design [-quiet] [-verbose] [<name>]
```

Returns

TCL_OK, TCL_ERROR if failed.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<name>]	Name of design to save.

Categories

[IPIntegrator](#)

Description

Saves any changes to the current or specified IP subsystem design in the IP integrator of the Vivado Design Suite.

This command returns TCL_OK if it is successful, or TCL_ERROR if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - Specify the name of the IP subsystem design to save. If name is not specified, the current IP subsystem design will be saved.

Send Feedback

Examples

The following example saves the current IP subsystem design in the current project:

```
save_bd_design
```

See Also

- close_bd_design
- create_bd_design
- current_bd_design
- get_bd_designs
- open_bd_design

save_bd_design_as

Save a copy of the existing IP subsystem design to specified disk file with a different name. Generated output products will not be saved.

Syntax

```
save_bd_design_as [-dir <arg>] [-ignore_comments] [-freeze_boundary]
    [-force] [-quiet] [-verbose] [<name>]
```

Returns

TCL_OK, TCL_ERROR if failed.

Usage

Name	Description
`[-dir]`	Directory path for remote BD to be created and managed. This is required if a name is not specified
`[-ignore_comments]`	Do not save user comments
`[-freeze_boundary]`	Freeze the boundary for the new design
`[-force]`	Overwrite existing file if present
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`[<name>]`	Name of the design to create. This is required if a directory is not specified

Categories

IPIntegrator

Description

Save a copy of an existing block design from the IP integrator to a new location, or with a different name. The generated output products of the block design will not be saved to the new block design.

Note: You cannot create a copy of a block design that has locked IP. The IP must be unlocked or the command will return an error.

Arguments

```
save_bd_design_as  [-dir <arg>] [-ignore_comments] [-force] [-quiet] [-
verbose]
                   [<name>]
```

Send Feedback

`-dir <arg>` - (Optional) The directory into which the new block design files are saved. The block design will be saved to the specified location, in a folder with same name as the block design. If you specify just the `-dir` option, the block design will be copied to the new location with the current name.

> ⭐ **IMPORTANT!** *Although `-dir` is optional, either the `-dir` or a `<name>` must be specified.*

`-ignore_comments` - (Optional) Do not copy any user-defined comment blocks in the current block design.

`-force` - (Optional) Overwrite an existing block design of the same name if one exists.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - Specify the name of the block design to save. If `<name>` is not specified, the `-dir` option must be specified. If you specify `<name>` without `-dir`, the block design will be saved to its current location with the new name.

Examples

The following example saves the current block design to the specified directory, and renames it to the specified name:

```
save_bd_design_as -dir C:/Data new_Block
```

See Also

- close_bd_design
- create_bd_design
- current_bd_design
- get_bd_designs
- open_bd_design
- save_bd_design

save_constraints

Save the current design's constraints

Syntax

```
save_constraints [-force] [-quiet] [-verbose]
```

Usage

Name	Description
[-force]	Force constraints save, overwriting the target and source XDC if necessary
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Project

Description

Saves any changes to the constraints files of the active constraints set. This command writes any changes to the constraints files to the project data on the hard drive; saving any work in progress and committing any changes.

Arguments

`-force` - (Optional) Save the active constraints files regardless of whether any changes have been made, overwriting the current target constraints file.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example saves the constraints files for the active constraints set regardless of any changes to the files:

```
save_constraints -force
```

See Also

- save_constraints_as

save_constraints_as

Save current design's constraints as a new set of constraints files

Syntax

```
save_constraints_as [-dir <arg>] [-target_constrs_file <arg>] [-quiet]
    [-verbose] <name>
```

Usage

Name	Description
[-dir]	Directory to save constraints to
[-target_constrs_file]	Target constraints file for the new fileset
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<name>	Name of the new constraints fileset

Categories

Project

Description

Copies the active constraints set to create a new constraints set, with local copies of any constraints files that are part of the constraints set. You can also specify a new constraints file to use as the target for the copied constraints set.

Use this command to save changes to the constraints in a design without affecting the current constraints files. This allows you to do some "what-if" type development of design constraints.

Note: The new constraint set created by the `save_constraints_as` command will not be active in the design, although it will be referenced by the design. To make the constraints set active you must set the constrset property to point to the new constraints set for specific runs. See the example below.

Arguments

`-dir <arg>` - (Optional) The directory into which constraints files are saved. If the directory is not specified, the new constraints set is located in the project sources directory. The constraints files from the active constraints set are copied into the specified directory.

`-target_constrs_file <arg>` - (Optional) Specifies a new target constraints file for the new constraints fileset. If a path is not specified as part of the file name, the file will be created in the fileset directory.

Note: You must specify the `.xdc` file extension, or the command will report a warning that the filetype is invalid, and cannot be set to the target constraint set. In this case, the existing target constraints file will be used as the target.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - (Required) The name of the constraints set to write.

Examples

The following example saves the active constraints set into a new constraints set called constrs_2, and copies any constraints files into the specified directory, as well as creating a new target constraints file for the constraints set:

```
save_constraints_as -dir C:/Data/con1 \
    -target_constrs_file rev1.xdc constrs_2
```

The following example saves the active constraints set as a new constraints set called newCon2, and copies any constraint files into the newCon2 constraint directory under project sources. The constrset property for the specified synthesis and implementation runs are then set to point to the new constraints set:

```
save_constraints_as newCon2
set_property CONSTRSET newCon2 [get_runs synth_1]
set_property CONSTRSET newCon2 [get_runs impl_1]
```

Note: The constraints set is not active in the design until it has been set to active for the current runs.

See Also

- save_constraints

save_project_as

Save the current project under a new name

Syntax

```
save_project_as [-scan_for_includes] [-exclude_run_results]
    [-include_local_ip_cache] [-force] [-quiet] [-verbose] <name> [<dir>]
```

Returns

Saved project object.

Usage

Name	Description
[-scan_for_includes]	Scan for include files and add them to the new project
[-exclude_run_results]	Exclude run results in the new project
[-include_local_ip_cache]	Include IP cache results in the new project
[-force]	Overwrite existing project directory
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<name>	New name for the project to save
[<dir>]	Directory where the project file is saved Default: .

Categories

Project

Description

Saves a currently open project file under a new name in the specified directory, or in the current working directory if no other directory is specified.

This command save a Vivado Design Suite project file (.xpr), or a project file for the Vivado Lab Edition (.lpr), in the specified directory.

The command returns the name of the saved project, or returns an error if it fails.

Arguments

-scan_for_includes - (Optional) Scans all source files and adds any referenced Verilog 'include files into the project structure.

Send Feedback

`-exclude_run_results` - (Optional) Exclude run results in the new project. This strips the project of its current run results to make a copy of the project sources without the additional data.

`-include_local_ip_cache` - (Optional) Include IP cache results in the new project. This allows you to retain the out-of-context synthesis cache with the new project.

`-force` - (Optional) Overwrite the existing project. If the project name is already define in the specified directory then you must also specify the `-force` option for the tool to overwrite the existing project.

Note: If the existing project is currently open, the new project will overwrite the existing project on the disk, but both projects will be opened in the tool. In this case you should probably run the `close_project` command prior to running `create_project`.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - (Required) The name of the new project. This argument must appear before the specified directory. Since these commands do not have parameters, the tool interprets the first argument as `<name>` and uses the second argument as `<dir>`. The project file is saved as `<name>.xpr` in the Vivado Design Suite, or `<name>.lpr` in the Vivado Lab Edition, and is written into the specified directory <dir>.

`<dir>` - (Optional) The directory name in which to write the new project file. If the specified directory does not exist a new directory will be created. If the directory is specified with the complete path, the tool uses the specified path name. However, if <dir> is specified without a path, the tool looks for or creates the directory in the current working directory, or the directory from which the tool was launched.

Note: When creating a project in GUI-mode, the tool appends the filename `<name>` to the directory name `<dir>` and creates a project directory with the name `<dir>/<name>` and places the new project file and project data folder into that project directory.

Examples

The following example saves the active project as a new project called myProject in a directory called myProjectDir:

```
save_project_as myProject myProjectDir
```

Note: Because <dir> is specified as the folder name only, the tool will create the project in the current working directory, or the directory from which the tool was launched.

The following example saves the current project to a new project called myProject in a directory called C:/Designs/myProjectDir. If you use the `-force` argument, the tool will overwrite an existing project if one is found in the specified location.

```
save_project_as myProject C:/Designs/myProjectDir -force
```

See Also

- create_project
- current_project
- open_project

save_wave_config

Saves the specified or current wave configuration object to the given filename

Syntax

```
save_wave_config [-object <args>] [-quiet] [-verbose] [<filename>]
```

Returns

The wave configuration object saved.

Usage

Name	Description
[-object]	The WCFG or wave configuration to save. Default: Current wave configuration
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<filename>]	Filename to save the specified or current wave configuration object

Categories

Waveform

Description

Save the current or specified wave configuration object to a specified filename.

If the wave configuration object has not been saved before, and does not have a FILE_PATH property value, the <filename> is required and the NAME of the wave configuration object will be changed to match the specified <filename>.

If the specified wave configuration object has been previously saved, and has a FILE_PATH property, the object will be written to its current location, and the <filename> does not need to be specified.

If the wave configuration object has a FILE_PATH property, but a different <filename> is specified, the wave configuration object will be saved to the new <filename>, and the object will be renamed to match the specified <filename>.

Send Feedback

Arguments

`-object <arg>` - (Optional) Specify a wave configuration object to save. If the wave configuration object has not been saved to a file before, you must also specify the <filename>. If `-object` is not specified, the current wave configuration object, as returned by `current_wave_config` is saved to the specified <filename>.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<filename> - (Optional) Specify a path and <filename> to save the current or specified wave configuration object. The <filename> should have a suffix of `.wcfg`, and the file suffix will be assigned automatically if no other suffix is supplied.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

Examples

The following example saves the specified wave configuration object to a new filename:

```
save_wave_config -object [get_wave_configs test.wcfg] \
C:/Data/project/newTest
```

Note: The wave config file will be assigned the `.wcfg` suffix since none is specified.

See Also

- create_wave_config
- current_wave_config
- get_wave_configs
- open_wave_config

scan_dr_hw_jtag

Perform shift DR on 'hw_jtag'.

Syntax

```
scan_dr_hw_jtag [-tdi <arg>] [-tdo <arg>] [-mask <arg>] [-smask <arg>]
    [-quiet] [-verbose] <length>
```

Returns

Hardware TDO

Usage

Name	Description
[-tdi]	Hex value to be scanned into the target
[-tdo]	Hex value to be compared against the scanned value
[-mask]	Hex value mask applied when comparing TDO values
[-smask]	Hex value mask applied to TDI value
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<length>	Number of bits to be scanned.

Categories

[Hardware](), [Object]()

Description

The `scan_dr_hw_jtag` command specifies a scan pattern to be scanned into the JTAG interface target data register.

The command targets a hw_jtag object which is created when the hw_target is opened in JTAG mode through the use of the `open_hw_target -jtag_mode` command.

When targeting the hw_jtag object prior to shifting the scan pattern specified in the `scan_dr_hw_jtag` command, the last defined header property (HDR) will be pre-pended to the beginning of the specified data pattern, and the last defined trailer property (TDR) will be appended to the end of the data pattern.

The options can be specified in any order, but can only be specified once. The number of bits represented by the hex strings specified for `-tdi`, `-tdo`, `-mask`, or `-smask` cannot be greater than the maximum specified by <length>. Leading zeros are assumed for a hex string if the number of bits represented by the hex strings specified is less than the <length>.

Send Feedback

When shifting the data bits to the target data register, the `scan_dr_hw_jtag` command moves the JTAG TAP from the current stable state to the DRSHIFT state according to the state transition table below:

```
Current State       Transitions to get to DRSHIFT state
RESET               IDLE -> DRSELECT -> DRCAPTURE -> DRSHIFT
IDLE                DRSELECT -> DRCAPTURE -> DRSHIFT
IRPAUSE             IREXIT2 -> IRUPDATE -> DRSELECT -> DRCAPTURE -> DRSHIFT
DRPAUSE             DREXIT2 -> DRSHIFT
DRPAUSE*            DREXIT2 -> DRUPDATE -> DRSELECT -> DRCAPTURE -> DRSHIFT
```

Note: * With `-force_update` option set.

After the last data bit is shifted into the target data register, the `scan_dr_hw_jtag` command moves the JTAG TAP to the IDLE state, or to the stable state defined by the `run_state_hw_jtag` command.

The `scan_dr_hw_jtag` command returns a hex array containing captured TDO data from the hw_jtag, or returns an error if it fails.

The command raises an error that can be trapped by the Tcl `catch` command if TDO data from the hw_jtag does not match specified `-tdo` argument.

> **TIP:** *If `-tdo` and `-mask` arguments are specified, then the mask is applied to the `-tdo` option and the hw_jtag TDO data returned before comparing the two.*

Arguments

`-tdi <arg>` - (Optional) The value to be scanned into the target, expressed as a hex value. If this option is not specified, the `-tdi` value from the last `scan_dr_hw_jtag` command will be used. The `-tdi` option must be explicitly specified for the first `scan_dr_hw_jtag` command, and when the <length> changes.

`-tdo <arg>` - (Optional) Specifies the data value, expressed as a hex string, to be compared against the actual TDO value scanned out of the hw_jtag data register. If this option is not specified no comparison will be performed. If the `-tdo` option is not specified, the `-mask` option will be ignored.

`-mask <arg>` - (Optional) The mask to use when comparing `-tdo` value against the actual TDO value scanned out of the hw_jtag. A '1' in a specific bit position indicates the bit value should be compared. A '0' indicates the value should not be used for comparison. If `-mask` is not specified, the `-mask` value from the last `scan_dr_hw_jtag` command will be used.

> **IMPORTANT!** *If the <length> changes and the `-mask` option is not specified, the `-mask` pattern used is all '1's.*

Send Feedback

`-smask <arg>` - (Optional) The mask to use with `-tdi` data. A '1' in a specific bit position indicates the TDI data in that bit position is significant; a '0' indicates it is not. The `-smask` option will be applied even if the `-tdi` option is not specified. If `-smask` is not specified, the `-smask` value from the last `scan_dr_hw_jtag` command will be used.

> ⭐ **IMPORTANT!** *If the <length> changes and the `-smask` option is not specified, the `-smask` pattern used is all '1's.*

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<length> - (Required) A 32-bit unsigned decimal integer greater than 0, specifying the number of bits to be scanned from the data register.

Example

The following example scans the JTAG data register for a 24 bit value:

```
scan_dr_hw_jtag  24
```

The following example sends a 24 bit value 0x00_0010 (LSB first) to TDI, then captures the data output, TDO, applies a mask with 0xF3_FFFF, and compares the returned TDO value against the specified value -tdo 0x81_8181:

```
scan_dr_hw_jtag  24 -tdi 000010 -tdo 818181 -mask F3FFFF -smask 0
```

This example pads the specified TDI value 0x3f with leading 0x:

```
scan_dr_hw_jtag 24 -tdi 3f
```

To break up a long data register shift into multiple SDR shifts, specify an end_state of DRPAUSE. This will cause the first `scan_dr_hw_jtag` command to end in the DRPAUSE stable state, and then the subsequent `scan_dr_hw_jtag` commands will go to DREXIT2 state before going back to DRSHIFT.

```
run_state_hw_jtag DRPAUSE
scan_dr_hw_jtag 16 -tdi aabb
scan_dr_hw_jtag 16 -tdi ccdd
scan_dr_hw_jtag 16 -tdi ceff
scan_ir_hw_jtag 6 -tdi 05
```

See Also

- connect_hw_server
- current_hw_target
- open_hw_target
- run_state_hw_jtag
- runtest_hw_jtag
- scan_ir_hw_jtag

scan_ir_hw_jtag

Perform shift IR on 'hw_jtag'.

Syntax

```
scan_ir_hw_jtag [-tdi <arg>] [-tdo <arg>] [-mask <arg>] [-smask <arg>]
      [-quiet] [-verbose] <length>
```

Returns

Hardware TDO

Usage

Name	Description
[-tdi]	Hex value to be scanned into the target
[-tdo]	Hex value to be compared against the scanned value
[-mask]	Hex value mask applied when comparing TDO values
[-smask]	Hex value mask applied to TDI value
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<length>	Number of bits to be scanned.

Categories

[Hardware](), [Object]()

Description

The `scan_ir_hw_jtag` command specifies a scan pattern to be scanned into the JTAG interface target instruction register.

The command targets a hw_jtag object which is created when the hw_target is opened in JTAG mode through the use of the `open_hw_target -jtag_mode` command.

When targeting the hw_jtag object prior to shifting the scan pattern specified in the `scan_ir_hw_jtag` command, the last defined header property (HIR) will be pre-pended to the beginning of the specified data pattern, and the last defined trailer property (TIR) will be appended to the end of the data pattern.

The options can be specified in any order, but can only be specified once. The number of bits represented by the hex strings specified for `-tdi`, `-tdo`, `-mask`, or `-smask` cannot be greater than the maximum specified by <length>. Leading zeros are assumed for a hex string if the number of bits represented by the hex strings specified is less than the <length>.

Send Feedback

When shifting the bits into the target instruction register, the `scan_ir_hw_jtag` command moves the JTAG TAP from the current stable state to the IRSHIFT state according to the state transition table below:

```
Current      Transitions to get to
State        IRSHIFT state
RESET        IDLE > DRSELECT > IRSELECT > IRCAPTURE > IRSHIFT
IDLE         IRSELECT > IRCAPTURE > IRSHIFT
DRPAUSE      DREXIT2 > DRUPDATE > DRSELECT > IRSELECT > IRCAPTURE > IRSHIFT
IRPAUSE      IREXIT2 > IRSHIFT
IRPAUSE*     IREXIT2 > IRUPDATE > DRSELECT > IRSELECT > IRCAPTURE > IRSHIFT
```

Note: * With `-force_update` option set.

After the last data bit is shifted into the target data register, the `scan_ir_hw_jtag` command moves the JTAG TAP to the IDLE state, or to the stable state defined by the run_state_hw_jtag command.

The `scan_ir_hw_jtag` command returns a hex array containing captured TDO data from the hw_jtag, or returns an error if it fails.

The command raises an error that can be trapped by the Tcl `catch` command if TDO data from the hw_jtag does not match specified `-tdo` argument.

> **TIP:** *If `-tdo` and `-mask` arguments are specified, then the mask is applied to the `-tdo` option and the hw_jtag TDO data returned before comparing the two.*

Arguments

`-tdi <arg>` - (Optional) The value to be scanned into the target, expressed as a hex value. If this option is not specified, the `-tdi` value from the last `scan_ir_hw_jtag` command will be used. The `-tdi` option must be explicitly specified for the first `scan_ir_hw_jtag` command, and when the <length> changes.

`-tdo <arg>` - (Optional) Specifies the data value, expressed as a hex string, to be compared against the actual TDO value scanned out of the hw_jtag instruction register. If this option is not specified no comparison will be performed. If the `-tdo` option is not specified, the `-mask` option will be ignored.

`-mask <arg>` - (Optional) The mask to use when comparing `-tdo` value against the actual TDO value scanned out of the hw_jtag. A '1' in a specific bit position indicates the bit value should be compared. A '0' indicates the value should not be used for comparison. If `-mask` is not specified, the `-mask` value from the last `scan_ir_hw_jtag` command will be used.

> **IMPORTANT!** *If the <length> changes and the `-mask` option is not specified, the `-mask` pattern used is all '1's.*

`-smask <arg>` - (Optional) The mask to use with `-tdi` data. A '1' in a specific bit position indicates the TDI data in that bit position is significant; a '0' indicates it is not. The `-smask` option will be applied even if the `-tdi` option is not specified. If `-smask` is not specified, the `-smask` value from the last `scan_ir_hw_jtag` command will be used.

> ⭐ **IMPORTANT!** *If the <length> changes and the `-smask` option is not specified, the `-smask` pattern used is all '1's.*

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<length> - (Required) A 32-bit unsigned decimal integer greater than 0, specifying the number of bits to be scanned from the instruction register.

Example

The following example scans the JTAG instruction register for a 24 bit value:

```
scan_ir_hw_jtag  24
```

The following example sends a 24 bit value 0x00_0010 (LSB first) to TDI, then captures the TDO output, applies a mask with 0xF3_FFFF, and compares the returned TDO value against the specified value -tdo 0x81_8181:

```
scan_ir_hw_jtag  24 -tdi 000010 -tdo 818181 -mask F3FFFF -smask 0
```

This example pads the specified TDI value 0x3f with leading 0x:

```
scan_ir_hw_jtag 24 -tdi 3f
```

To break up a long instruction register shift into multiple shifts, specify an end_state of IRPAUSE. This will cause the first `scan_ir_hw_jtag` command to end in the IRPAUSE stable state, and then the subsequent `scan_ir_hw_jtag` commands will go to IREXIT2 state before going back to IRSHIFT.

```
run_state_hw_jtag IRPAUSE
scan_ir_hw_jtag 8 -tdi aa
scan_ir_hw_jtag 8 -tdi bb
scan_ir_hw_jtag 8 -tdi cc
scan_dr_hw_jtag 128 -tdi 0
```

Send Feedback

See Also

- connect_hw_server
- current_hw_target
- open_hw_target
- run_state_hw_jtag
- runtest_hw_jtag

select_objects

Select objects in GUI

Syntax

```
select_objects [-add] [-quiet] [-verbose] <objects>
```

Usage

Name	Description
[-add]	Add to existing selection list
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<objects>	Objects to select

Categories

GUIControl

Description

Selects the specified object in the appropriate open views in the GUI mode. This command is for display purposes only. You must use the `get_selected_objects` command to return the selected objects for use in other commands.

The `select_objects` command may select secondary objects in addition to the primary object specified. The selection of secondary objects is controlled through the use of Selection Rules defined in the **Tools → Settings** command. Refer to the *Vivado Design Suite User Guide: Using the IDE (UG893)* for more information on Setting Selection Rules.

Selected objects can be unselected with the `unselect_objects` command.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<objects>` - (Required) Specifies one or more objects to be selected.

Examples

The following example selects the specified site on the device:

```
select_objects [get_sites SLICE_X56Y214]
```

See Also

- get_selected_objects
- highlight_objects
- mark_objects
- unselect_objects

select_wave_objects

Display help for one or more topics

Syntax

```
select_wave_objects [-quiet] [-verbose] <items>...
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<items>	select waveform objects

Categories

Waveform

Description

Selects the specified object in the Waveform window of the Vivado IDE. This command is for selecting displayed items in the Waveform window only, and is similar to the select_objects command in the Vivado IDE.

Note: Use the `get_hdl_objects` command to select simulation objects in the open simulation, or current_sim.

Unselect selected objects using the `select_wave_objects` command with an empty string:

```
select_wave_objects ""
```

This command returns nothing if successful, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<items> - (Required) Specifies one or more items in the Waveform window to be selected. Items are specified by name. Specifying an empty string unselects all currently selected waveform objects.

Examples

The following example selects the specified site on the device:

```
select_wave_objects {sys_clk_p sysc_clk_n}
```

See Also

- get_value
- select_objects
- unselect_objects

Send Feedback

set_bus_skew

Define bus skew

Syntax

```
set_bus_skew [-from <args>] [-rise_from <args>] [-fall_from <args>]
    [-to <args>] [-rise_to <args>] [-fall_to <args>] [-through <args>]
    [-rise_through <args>] [-fall_through <args>] [-quiet] [-verbose]
    <value>
```

Usage

Name	Description
[-from]	List of path startpoints or clocks
[-rise_from]	Apply to paths rising from the list of startpoints or clocks
[-fall_from]	Apply to paths falling from the list of startpoints or clocks
[-to]	List of path endpoints or clocks
[-rise_to]	Apply to paths with rise transition at the list of endpoints or clocks
[-fall_to]	Apply to paths with fall transition at the list of endpoints or clocks
[-through]	List of through pins, cells or nets
[-rise_through]	Apply to paths rising through pins, cells or nets
[-fall_through]	Apply to paths falling through pins, cells or nets
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<value>	Constraint value

Categories

[XDC](XDC)

Description

Set the bus skew requirement on bus signals that cross clock domains. The bus skew constraint defines the maximum skew spread between the fastest and slowest signals of the bus, and does not consider the overall datapath delay. The Vivado router will try to satisfy the `set_bus_skew` constraints. Example uses of the bus skew constraint include clock domain crossing for gray-coded pointers, MUX-controlled and MUX-data holding CDC buses.

> 💡 **TIP:** *Bus skew constraints are not overridden by clock groups, max delay, or false path, because* `set_bus_skew` *is a constraint between the signals of a bus, rather than on a particular path.*

The `set_bus_skew` constraint can be combined with the `set_max_delay` constraint for good results. The `set_bus_skew` constraint does not care about the absolute datapath delay, but only about the relative arrival times of data at the destination, taking into account source and destination clock skew. You can help `set_bus_skew` by also using `set_max_delay -datapath_only <SRC_CLK>`. This constraint helps the Vivado placer to ensure that the source and destination registers are not placed too far apart, so that the router can more easily satisfy the `set_bus_skew` constraint. Refer to the *Vivado Design Suite User Guide: Using Constraints* (UG903) for more information.

In order to not over constrain the skew requirement, the bus skew value should be approximately the smallest period of the two clock domains. This will prevent multiple data captures by the destination clock domain.

The `set_bus_skew` command requires a timing path defined by both `-from` and `-to`, or some form such as `-fall_from` or `-rise_to`. You can optionally specify `-through` values to further refine the path. You should specify explicit signal paths with `-from/-to` instead of specifying entire clock domains:

- `set_bus_skew -from [get_pins <hierarchy/C>] -to [get_pins <hierarchy/D>] <value>`

- `set_bus_skew -from [get_clocks <clock name>] -to get_clocks <clock name>] <value>`

> 💡 **TIP:** *Do not set bus skew constraints between timed synchronous clock domains.*

You can use the `report_bus_skew` command to report the calculated skew on paths in the current design.

The `set_bus_skew` command returns nothing if successful, or an error if it fails.

Arguments

`-from` <args> - (Optional) The starting points of the timing paths to set bus skew on. Pins or cells can be specified as timing path startpoints. You can also specify a clock object, and all startpoints clocked by the named clock will be analyzed.

> ⭐ **IMPORTANT!** *Although the various `-from`, `-to`, and `-through` arguments are listed as optional, a path must be defined using a form of the `-from` and `-to` arguments.*

`-rise_from` <args> - (Optional) Set bus skew on the timing paths with rising edge transitions at the specified startpoints. If a clock object is specified, only the paths launched by the rising edge of the clock are considered as startpoints.

`-fall_from` <args> - (Optional) Set bus skew on the timing paths with falling edge transitions at the specified startpoints. If a clock object is specified, only the paths launched by the falling edge of the clock are considered as startpoints.

`-to` <args> - (Optional) The endpoints, or destination objects of timing paths to define the bus skew on. Pins or cells can be specified as endpoints. A clock object can also be specified, in which case endpoints clocked by the named clock are analyzed.

`-rise_to` <args> - (Optional) Set bus skew on the timing paths with rising edge transitions at the specified endpoints. If a clock object is specified, only the paths captured by the rising edge of the named clock are considered as endpoints.

`-fall_to` <args> - (Optional) Set bus skew on the timing paths with falling edge transitions at the specified endpoints. If a clock object is specified, only the paths captured by the falling edge of the named clock are considered as endpoints.

`-through` <args> - (Optional) Set bus skew only on paths through the specified pins, cell instance, or nets. You can specify individual `-through` (or `-rise_through` and `-fall_through`) points in sequence to define a specific path through the design. The order of the through points is important to define a path. You can also specify through points with multiple objects, in which case the timing path can pass through any of the specified through objects.

`-rise_through` <args> - (Optional) Set bus skew on the timing paths with rising edge transitions at the specified through points.

`-fall_through` <args> - (Optional) Set bus skew on the timing paths with falling edge transitions at the specified through points.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<value>` - Specifies the value of the acceptable bus skew in nanoseconds.

Examples

The following example defines the bus skew between the gray-coded Read and Write pointers:

```
set_bus_skew -from [get_pins gray_coded_read_ptr[*]/C] \
    -to [get_pins gray_coded_write_ptr[*]/D] 2.5
```

See Also

- report_bus_skew
- report_cdc
- report_datasheet
- set_data_check
- set_max_delay

set_case_analysis

Specify that an input is 1, 0, rising or falling

Syntax

```
set_case_analysis [-quiet] [-verbose] <value> <objects>
```

Usage

Name	Description
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<value>`	Logic value on the pin: Values: 0, 1, rising, falling, zero, one, rise, fall
`<objects>`	List of ports or pins

Categories

[SDC](#), [XDC](#)

Description

Specifies that a pin or port is in a steady state of 1, 0, rising or falling.

This command is usually used to force values onto the ports to help reduce the analysis space, runtime and memory consumption. It is important to let the Vivado timing engine know about signals that have a constant value. This is also critical to ensure that non-functional and irrelevant paths are not reported.

Setting a case value on a pin results in disabling timing analysis through that pin. This means that timing paths through that pin are not reported.

Note: This command returns nothing if successful, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<value>` - (Required) The value to use on the port or pin for timing analysis. The valid values are 0 or zero, 1 or one, rise or rising, fall or falling. When the values ris(e)(ing) or fall(ing) are specified, the given pins or ports are only considered for timing analysis with the specified transition. The other transition is disabled. The default setting is 1.

`<objects>` - (Required) One or more ports or pins on which to apply the `<value>`.

Examples

In the example below, two clocks are created on the input pins of the BUFGMUX, clock_sel, but only clk_B is propagated through the output pin after setting the constant value 1 on the selection pin S:

```
create_clock -name clk_A -period 10.0 [get_pins clock_sel/I0]
create_clock -name clk_B -period 15.0 [get_pins clock_sel/I1]
set_case_analysis 1 [get_pins clock_sel/S]
```

See Also

- create_clock
- report_timing

set_clock_groups

Set exclusive or asynchronous clock groups

Syntax

```
set_clock_groups [-name <arg>] [-logically_exclusive]
    [-physically_exclusive] [-asynchronous] [-group <args>] [-quiet]
    [-verbose]
```

Usage

Name	Description
[-name]	Name for clock grouping
[-logically_exclusive]	Specify logically exclusive clock groups
[-physically_exclusive]	Specify physically exclusive clock groups
[-asynchronous]	Specify asynchronous clock groups
[-group]	Clocks List
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[SDC](), [XDC]()

Description

> 💡 **TIP:** *The XDC > Timing Constraints language templates and the Timing Constraints Wizard in the Vivado IDE offer timing diagrams and additional details around defining specific timing constraints. You can refer to these sources for additional information.*

Define clocks, or groups of clocks, that are exclusive with or asynchronous to other clocks in the design. Exclusive clocks are not active at the same time, and paths between them can be ignored during timing analysis. Asynchronous clocks are clocks with no known phase relationship, which typically happens when they do not share the same primary clock or do not have a common period.

Using this command is similar to defining false path constraints for data paths moving between exclusive or asynchronous clock domains. See the *Vivado Design Suite User Guide: Using Constraints* (UG903) for more information.

If only one group is specified, the clocks in that group are asynchronous or exclusive to all other clocks in the design, but not to each other. If a new clock is created after the `set_clock_groups` command, it is asynchronous to that group as well.

This command can also be used for multiple clocks that are derived from a single BUFGMUX as both of the clocks will not be active at the same time.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

`-name` <group_name> - (Optional) Name of the clock group to be created. A name will be automatically assigned if one is not specified.

`-logically_exclusive` - (Optional) The specified clocks are logically exclusive.

Note: `-logically_exclusive`, `-physically_exclusive` and `-asynchronous` are mutually exclusive arguments.

`-physically_exclusive` - (Optional) The specified clocks are physically exclusive, and cannot exist in the design at the same time.

`-asynchronous` - (Optional) The specified clocks are asynchronous to one another.

`-group` <args> - (Optional) The list of clocks to be included in the clock group. Each group of clocks is exclusive with or asynchronous with the clocks specified in all other groups.

💡 **TIP:** *If only one group of clocks is specified, that group is exclusive with or asynchronous to all other clocks in the design. Clocks can be specified by name, or as clock objects returned by the* `get_clocks` *command.*

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

Group all the elements driven by src_clk and sync_clk into separate clock groups. The clock groups are asynchronous to each other:

```
set_clock_groups -group src_clk -group sync_clk -asynchronous
```

The following example includes the generated clocks of the specified clocks, and adds those to the clock group:

```
set_clock_groups -group [get_clocks -include_generated_clocks src_clk] \
-group [get_clocks -include_generated_clocks sync_clk]  -asynchronous
```

Note: In the preceding example, src_clk and sync_clk, and all their generated clocks, are asynchronous. Otherwise the generated clocks would be timed against each other and the other master clock.

In this example, the specified clocks are grouped together, and are asynchronous to all other clocks in the design:

```
set_clock_groups -async -group [get_clocks {J_CLK U_CLK}]
```

See Also

- get_clocks
- set_false_path

Send Feedback

set_clock_latency

Capture actual or predicted clock latency

Syntax

```
set_clock_latency [-clock <args>] [-rise] [-fall] [-min] [-max] [-source]
     [-late] [-early] [-quiet] [-verbose] <latency> <objects>
```

Usage

Name	Description
[-clock]	List of relative clocks
[-rise]	Specify clock rise latency
[-fall]	Specify clock fall latency
[-min]	Specify clock rise and fall min condition latency
[-max]	Specify clock rise and fall max condition latency
[-source]	Specify clock rise and fall source latency
[-late]	Specify clock rise and fall late source latency
[-early]	Specify clock rise and fall early source latency
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<latency>	Latency value
<objects>	List of clocks, ports or pins

Categories

SDC, XDC

Description

This command defines a clock's source or network latency for specified clocks, ports, or pins.

Note: This command operates silently and does not return direct feedback of its operation.

Source latency is the time in nanoseconds that a clock signal takes to propagate from its waveform origin to the clock definition point in the design. For example, this would be the time delay for the clock to propagate from its source (oscillator) on the system board to the FPGA input port.

Network latency is the time a clock signal takes to propagate from its definition point in the design to a register clock pin on the timing path. The total clock latency at a register clock pin is the sum of a clock's source latency and network latency.

Arguments

`-clock` <args> - (Optional) Specifies a list of clocks associated with the `<latency>` assigned to the specified `<objects>`. If the `-clock` argument is not used, the clock `<latency>` will be applied to all clocks passing through the specified pins and ports.

`-rise` - (Optional) Defines the latency for the rising clock edge.

`-fall` - (Optional) Defines the latency for the falling clock edge.

`-min` - (Optional) Defines the minimum latency for the specified clocks for multi-corner analysis.

`-max` - (Optional) Defines the maximum latency for the specified clocks for multi-corner analysis.

Note: The `-min` and `-max` options are mutually exclusive.

`-source` - (Optional) Defines the specified `<latency>` as a source latency. Clock source latencies can only be specified for clock objects and clock source pins.

Note: Without the `-source` argument the `<latency>` is considered as network latency.

`-late` - (Optional) The time delay specified by `-latency` is how late the clock edge arrives.

`-early` - (Optional) The time delay specified by `-latency` is how early the clock edge arrives.

Note: The `-early` and `-late` options are mutually exclusive, and can only be specified when `-source` is also specified.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<latency>` - (Required) The amount of clock latency, specified as nanoseconds, to apply.

`<objects>` - (Required) The clock, port, or pin objects on which to apply the latency. Specifying pin or port objects assigns the latency to all register clock pins in the transitive fanout of the pins or ports. If `-clock` is used, the latency is applied to all register clock pins of the specified clocks.

Note: If `<objects>` specifies a clock, the `-clock` argument is unnecessary, and is ignored.

Examples

This example will set an early latency on the rising edge of CLK_A.

```
set_clock_latency -source -rise -early 0.4 [get_ports CLK_A]
```

See Also

- report_timing

set_clock_sense

Set clock sense on ports or pins

Syntax

```
set_clock_sense [-positive] [-negative] [-stop_propagation]
    [-clocks <args>] [-quiet] [-verbose] <pins>
```

Usage

Name	Description
[-positive]	Specify positive unate (non_inverting) clock sense
[-negative]	Specify negative unate (inverting) clock sense
[-stop_propagation]	Stop clock propagation from specified pins
[-clocks]	List of clocks
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<pins>	List of port and/or pins

Categories

[SDC](), [XDC]()

Description

Sets clock sense at specified ports or pins. This is used to define the positive or negative unateness at the pin relative to a clock object. However, the specified unateness only applies at a non-unate point in the clock network, at a point where the clock signal cannot be determined. Since the clock signal is not determined, the defined clock sense propagates forward from the given pins.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

`-positive` - (Optional) The unate clock sense is positive (non_inverting).

`-negative` - (Optional) The unate clock sense is negative (inverting).

`-stop_propagation` - (Optional) Stop the propagation of clocks in the `-clocks` argument from the specified pins or ports. Propagation of the clock as clock and data is stopped.

Note: `-positive`, `-negative`, and `-stop_propagation` are mutually exclusive.

Send Feedback

`-clocks` <args> - (Optional) A list of clocks on which to apply the clock sense for the specified pins and ports. If the `-clocks` argument is not used, the clock sense will be applied to all clocks passing through the specified pins and ports.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<pins> - (Required) List of ports and pins to propagate the clock sense to.

Examples

The following example specifies that only the positive unate paths will propagate through the output pin of the XOR gate as compared with the original clock.

```
set_clock_sense -positive [get_pins xor_a.z]
```

See Also

- create_clock
- get_pins

set_clock_uncertainty

set clock uncertainty

Syntax

```
set_clock_uncertainty [-setup] [-hold] [-from <args>] [-rise_from <args>]
    [-fall_from <args>] [-to <args>] [-rise_to <args>] [-fall_to <args>]
    [-quiet] [-verbose] <uncertainty> [<objects>]
```

Usage

Name	Description
[-setup]	Specify clock uncertainty for setup checks
[-hold]	Specify clock uncertainty for hold checks
[-from]	Specify inter-clock uncertainty source clock
[-rise_from]	Specify inter-clock uncertainty source clock with rising edge
[-fall_from]	Specify inter-clock uncertainty source clock with falling edge
[-to]	Specify inter-clock uncertainty destination clock
[-rise_to]	Specify inter-clock uncertainty destination clock with rising edge
[-fall_to]	Specify inter-clock uncertainty destination clock with falling edge
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<uncertainty>	Uncertainty of clock network
[<objects>]	List of clocks, ports or pins

Categories

SDC, XDC

Description

This command is used to add to the uncertainty of a clock in the design, and does not override the default jitter calculation. This is referred to as the user clock uncertainty. The `set_clock_uncertainty` command provides a convenient means to over-constrain some clocks in the design without changing the clock definitions and relationships. It can constrain setup and hold paths separately using the `-setup` and `-hold` options.

Clock uncertainty is the maximum variation, specified in nanoseconds (ns), between two clock edges at registers within a single clock domain, or crossing between clock domains.

The clock uncertainty is used during setup and hold analysis, where uncertainty is calculated for each timing path based on the clock edges used by the analysis and the clock tree topology. For example, for a path where the startpoint and endpoint are connected to the same clock net, the clock uncertainty is null because the same clock edge is used for both source and destination, unless the `set_clock_uncertainty` command is used to add uncertainty for the min delay analysis. The Vivado timing engine uses clock uncertainty in the slack calculation as determined by the following equation:

- Setup Slack = Setup Path Requirement - Data Delay - Clock Uncertainty + Clock Skew

Clock Uncertainty is a function of different elements of jitter, as determined by the following equation which is returned by the `report_timing_summary` or `report_timing` commands:

- Clock Uncertainty = $(\sqrt{(T_{sj}^2 + D_j^2)})/2 + PE + UU$

Where:

- T_{sj} = Total System Jitter as calculated using the system jitter. See `set_system_jitter`.

- D_j = Discrete jitter is the amount of jitter introduced by hardware primitives such as MMCM or PLL. Discrete jitter is a feature of clocks generated by the MMCM, which includes the input jitter defined on the primary clock. See `set_input_jitter`.

- PE = Phase Error, which comes from the MMCM/PLL device model.

- UU = User Uncertainty, which defines the user clock uncertainty specified by this `set_clock_uncertainty` command.

> **TIP:** *SYSTEM_JITTER is reported as a property of clocks, although it applies to all clocks in the design. INPUT_JITTER is also a property of primary clocks. These properties can be returned by the* `get_property` *or* `report_property` *commands. Jitter and clock uncertainty are reported by the* `report_timing_summary` *and* `report_timing` *commands.*

This command returns nothing if successful, or returns an error if it fails.

Arguments

`-setup` - (Optional) The specified clock uncertainty is applied during setup checks.

`-hold` - (Optional) The specified clock uncertainty is applied during hold checks.

`-from` <source_clock_name> - (Optional) Specify inter-clock uncertainty source clock.

`-rise_from` <source_clock_name> - (Optional) Specify inter-clock uncertainty source clock with rising edge.

`-fall_from` <source_clock_name> - (Optional) Specify inter-clock uncertainty source clock with falling edge.

`-to` <destination_clock_name> - (Optional) Specify inter-clock uncertainty destination clock.

Send Feedback

`-rise_to` <destination_clock_name> - (Optional) Specify inter-clock uncertainty destination clock with rising edge.

`-fall_to` <destination_clock_name> - (Optional) Specify inter-clock uncertainty destination clock with falling edge.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<uncertainty>` - (Required) Uncertainty of the clock network, specified in nanoseconds.

`<objects>` - (Optional) List of clocks to define uncertainty for.

Examples

The following example defines the uncertainty between all clock domains:

```
set_clock_uncertainty 0.225 -from [get_clocks] -to [get_clocks]
```

The following command defines setup and hold uncertainty within the wbClk clock domain:

```
set_clock_uncertainty -setup 0.213 [get_clocks wbClk]
set_clock_uncertainty -hold 0.167 [get_clocks wbClk]
```

See Also

- create_clock
- create_generated_clock
- get_clocks
- set_input_jitter
- set_system_jitter

set_data_check

Create data to data checks

Syntax

```
set_data_check [-from <args>] [-to <args>] [-rise_from <args>]
    [-fall_from <args>] [-rise_to <args>] [-fall_to <args>] [-setup]
    [-hold] [-clock <args>] [-quiet] [-verbose] <value>
```

Usage

Name	Description
[-from]	From pin/port of data to data check
[-to]	To pin/port of the data to data check
[-rise_from]	Rise from pin/port of data to data check
[-fall_from]	Fall from pin/port of data to data check
[-rise_to]	Rise to pin/port of data to data check
[-fall_to]	Fall to pin/port of data to data check
[-setup]	Specify data check setup time
[-hold]	Specify data check hold time
[-clock]	Specify the clock domain at related pin/port of the checks
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<value>	Setup or hold time of the defined checks

Categories

[SDC](#), [XDC](#)

Description

Performs a setup and hold check for a data pin with respect to another data pin. This is different from a conventional setup and hold check that is done with respect to a clock pin.

This command defines min and max requirements between two endpoints, similar to setup (max) and hold (min) timing checks. Setup and hold checks are referenced from the related pin, specified by `-from`, to the constrained pin, specified by `-to`. The related pin is similar to the clock pin in a conventional setup and hold check. The timing analysis compares arrival times between the two specified endpoints. The difference must be less than the `set_data_check` `<value>` requirement in order to meet timing.

Limitations of the `set_data_check` command include:

- Variations in the destination clock delay are ignored.

- This command is used for timing purposes only, and is not considered by the Vivado placer or router.

Note: This command returns nothing if successful, or returns an error if it fails.

Arguments

`-from <value>` - (Optional) Check the datapath from the specified data pin, port, or net. The `-from` argument specifies the related pin.

`-to <value>` - (Optional) Check the datapath to the specified data pin, port, or net. The `-to` argument specifies the constrained pin

`-rise_from <value>` - (Optional) Check the datapath from the rising edge of the specified data pin, port, or net.

`-fall_from <value>` - (Optional) Check the datapath from the falling edge of the specified data pin, port, or net.

`-rise_to <value>` - (Optional) Check the datapath to the rising edge of the specified data pin, port, or net.

`-fall_to <value>` - (Optional) Check the datapath to the falling edge of the specified data pin, port, or net.

`-setup <value>` - (Optional) Perform only the setup data check. The default is to perform both setup and hold checks.

`-hold <value>` - (Optional) Perform the hold data check. The default is to perform both setup and hold checks.

`-clock <value>` - (Optional) Specify the clock domain at the related pin or port of the checks.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<value>` - (Required) The setup or hold time specified in nanoseconds (ns) for the defined data checks.

Examples

The following example defines a data check for a setup violation from pin A_IN to pin C_IN:

```
set_data_check -from A_IN -to C_IN -setup 2.0
```

In the above example, A_IN is the related pin and C_IN is the constrained pin. The above constraint would do a setup check of C_IN with respect to A_IN. The data at C_IN should arrive 2.0 ns prior to the edge of A_IN.

See Also

- report_timing
- set_min_delay
- set_max_delay

set_delay_model

Sets the interconnect delay model for timing analysis.

Syntax

```
set_delay_model [-interconnect <arg>] [-quiet] [-verbose]
```

Usage

Name	Description
[-interconnect]	Interconnect delay model used for timing analysis: Values: estimated, actual(default), none
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Timing

Description

Sets the interconnect delay model for timing analysis. There are three settings for the interconnect delay model: "actual", "estimated", or "none".

- If "actual" is selected, the actual delay from the routed interconnect will be used in timing analysis. If the design is only partially routed, then the actual delay from the routed portion will be used, along with estimated delay for the unrouted portion. The timing report will provide details regarding the source of the calculated delay.

- If "estimated" delays are selected, the timing analysis will include an estimate of the interconnect delays based on the placement and connectivity of the design onto the device prior to implementation. Estimated delay can be specified even if the design is fully routed.

- If "none" is selected, then no interconnect delay is included in the timing analysis, and only the logic delay is applied.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

`-interconnect [ actual | estimated | none ]` - (Optional) Delay model to be used. The default setting is `actual`.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following command will use a timing delay model which is an estimated value.

```
set_delay_model -interconnect estimated
```

See Also

• report_timing

Send Feedback

set_disable_timing

Disable timing arcs

Syntax

```
set_disable_timing [-from <arg>] [-to <arg>] [-quiet] [-verbose] <objects>
```

Usage

Name	Description
[-from]	From pin on cell
[-to]	To pin on cell
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<objects>	List of cells or pins, ports, lib-cells, lib-pins, libcell/cell timing-arcs

Categories

SDC, XDC, Timing

Description

Disables timing arcs within a specified cell or cells that lead to the output pins of the cell. Only the I/O paths between the clock port and the outputs of the cell are disabled.

The purpose of disabling a timing arc is to prevent timing analysis through the arc.

If a `<cell>` is specified, then all timing arcs in that cell are disabled. If the optional `-from` and `-to` arguments are specified, then the timing arcs are defined by the from/to pins. If only `-from` is speified then all timing arcs from that pin are disabled. If only `-to` is specified then all timing paths to that pin are disabled.

If a `<port>` is specified, then all timing paths from a specified input port are disabled, or timing paths to a specified output port are disabled.

Note: This command operates silently and does not return direct feedback of its operation

Arguments

`-from` <pin_name> - (Optional) Specifies the source pin of an object cell. The pin_name is specified by name only, without the need for the hierarchical cell name, which is defined by the `<object>`.

`-to <pin_name>` - (Optional) Specifies the destination pin of an object cell. The pin_name is specified by name only, without the need for the hierarchical cell name, which is defined by the `<object>`.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<objects>` - (Required) A list of one or more objects on which to disable the timing arcs. Must be specified as Vivado objects returned by `get_cells` or other appropriate Tcl commands. Can be any of the following types of objects: cells, ports, pins, lib-cells, lib-pins, lib-cell/cell timing arcs.

Examples

The following example disable the timing arc between the pins I0 and O of the LUT `div_dec_ff_i/U0/count_i_1` to break a combinational loop:

```
set_disable_timing -from I0 -to O [get_cells div_dec_ff_i/U0/count_i_1]
```

The following example disables the timing arcs between the specified input pin to the specified output pin of a BRAM cell:

```
set_disable_timing -from WEBWE[3] -to CLKMEM [get_cells \
    ldpc_dout360_channel/U_AP_FIFO_ldpc_dout360_channel_ram/mem_reg_0]
```

The following example disables all timing arcs of the specified cell:

```
set arcs [get_timing_arcs -of_objects [get_cells \
    ldpc_dout360_channel/U_AP_FIFO_ldpc_dout360_channel_ram/mem_reg_0]]
set_disable_timing $arcs
```

See Also

- get_cells
- get_timing_arcs
- report_timing

Send Feedback

set_external_delay

Set external delay

Syntax

```
set_external_delay -from <args> -to <args> [-min] [-max] [-add] [-quiet]
    [-verbose] <delay_value>
```

Usage

Name	Description
-from	Output port
-to	Input port
[-min]	Specifies minimum delay
[-max]	Specifies maximum delay
[-add]	Add to existing external delay
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<delay_value>	External (feedback) delay value

Categories

XDC, Timing

Description

> **TIP:** *The XDC > Timing Constraints language templates and the Timing Constraints Wizard in the Vivado IDE offer timing diagrams and additional details around defining specific timing constraints. You can refer to these sources for additional information.*

Sets the external (feedback) delay in nanoseconds (ns) between an output and input port. The external delay is used in the calculation of the PLL/MMCM compensation delay for PLLs/MMCMs with external feedback.

A min or max value can be specified. By default the value specified applies to both min (hold) and max (setup) compensation delays.

The command returns the defined delay.

Arguments

-from <arg> - (Required) The output port name.

-to <arg> - (Required) The input port name.

Send Feedback

`-min` - (Optional) Specifies the delay_value is a minimum delay value for hold time analysis.

`-max` - (Optional) Specifies the delay_value is a maximum delay value for setup analysis.

`-add` - (Optional) Add the specified delay to the existing external delay value.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<delay_value> - (Required) The external delay value specified as nanoseconds (ns). The default value is 0.

Examples

The following example sets the external feedback delay to 1.0 ns between the port ClkOut and ClkFb:

```
set_external_delay -from [get_ports ClkOut] -to [get_ports ClkFb] 1.0
```

See Also

- report_timing
- set_input_delay
- set_output_delay

Send Feedback

set_false_path

Define false path

Syntax

```
set_false_path [-setup] [-hold] [-rise] [-fall] [-reset_path]
    [-from <args>] [-rise_from <args>] [-fall_from <args>] [-to <args>]
    [-rise_to <args>] [-fall_to <args>] [-through <args>]
    [-rise_through <args>] [-fall_through <args>] [-quiet] [-verbose]
```

Usage

Name	Description
[-setup]	Eliminate setup timing analysis for paths
[-hold]	Eliminate hold timing analysis for paths
[-rise]	Eliminate only rising delays for the defined paths
[-fall]	Eliminate only falling delays for the defined paths
[-reset_path]	Reset this path before setting false path
[-from]	List of path startpoints or clocks
[-rise_from]	Apply to paths rising from the list of startpoints or clocks
[-fall_from]	Apply to paths falling from the list of startpoints or clocks
[-to]	List of path endpoints or clocks
[-rise_to]	Apply to paths with rise transition at the list of endpoints or clocks
[-fall_to]	Apply to paths with fall transition at the list of endpoints or clocks
[-through]	List of through pins, cells or nets
[-rise_through]	Apply to paths rising through pins, cells or nets
[-fall_through]	Apply to paths falling through pins, cells or nets
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[SDC](#), [XDC](#)

Description

> **TIP:** *The XDC > Timing Constraints language templates and the Timing Constraints Wizard in the Vivado IDE offer timing diagrams and additional details around defining specific timing constraints. You can refer to these sources for additional information.*

Sets false timing paths in the design that are ignored during timing analysis.

Note: This command operates silently and does not return direct feedback of its operation

Arguments

`-setup` - (Optional) Eliminate setup timing analysis for specified timing paths.

`-hold` - (Optional) Eliminate hold timing analysis for specified timing paths.

`-rise` - (Optional) Eliminate rising delays for the specified timing paths.

`-fall` - (Optional) Eliminate falling delays for the specified timing paths.

`-reset_path` - (Optional) Reset the timing path before setting false path. This clears all exception-based timing constraints from the defined timing path.

`-from` <element_name> - (Optional) List of path origins or clocks. A valid startpoint is a clock object, the clock pin of sequential logic, or an input or bidirectional port.

`-rise_from` <element_name> - (Optional) Apply to paths rising from the list of origins or clocks

`-fall_from` <element_name> - (Optional) Apply to paths falling from the list of origins or clocks

`-to` <element_name> - (Optional) List of path endpoints or clocks

`-rise_to` <element_name> - (Optional) Apply to paths with rise transition at the list of endpoints or clocks

`-fall_to` <element_name> - (Optional) Apply to paths with fall transition at the list of endpoints or clocks

`-through` <element_name> - (Optional) List of through pins, cells or nets

`-rise_through` <element_name> - (Optional) Apply to paths rising through pins, cells or nets

`-fall_through` <element_name> - (Optional) Apply to paths falling through pins, cells or nets

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example eliminates the setup timing for paths from the bftClk:

```
set_false_path -setup -from bftClk
```

The following example excludes paths between the two clocks from timing analysis:

```
set_false_path -from [get_clocks GT0_RXUSRCLK2_OUT] \
    -to [get_clocks DRPCLK_OUT]
```

See Also

- get_clocks
- get_pins
- get_ports
- report_timing

set_hierarchy_separator

Set hierarchical separator character

Syntax

```
set_hierarchy_separator [-quiet] [-verbose] [<separator>]
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<separator>]	Hierarchy separator character Default: /

Categories

SDC, XDC

Description

Sets the character that will be used for separating levels of hierarchy in the design.

Note: This command operates silently and does not return direct feedback of its operation

Arguments

<separator> - (Optional) The new character to use as a hierarchy separator. Valid characters to use as the hierarchy separator are: '/', '@', '^', '#', '.', and '|'. The default character is '/', and is used when no <separator> is specified.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

This example changes the hierarchy separator to the '|' character:

```
set_hierarchy_separator |
```

The following example restores the default hierarchy separator, '/':

```
set_hierarchy_separator
```

See Also

- get_hierarchy_separator

set_hw_sysmon_reg

Set the system monitor register value

Syntax

```
set_hw_sysmon_reg [-quiet] [-verbose] <hw_sysmon> <hexaddress> <hexdata>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hw_sysmon>	hw_sysmon object
<hexaddress>	Hex address to write to
<hexdata>	Hex write value

Categories

Hardware

Description

Set the system monitor register at the specified address to the hex value specified. This command identifies a register on the hw_sysmon on the current device through its hex address value, and sets the specified hex data value into that register.

> **IMPORTANT!** *Some of the registers on the system monitor are read-only and cannot be set directly. This command has no effect if you try to set the value of a read-only register on the system monitor.*

The System Monitor (SYSMON) Analog-to-Digital Converter (ADC) is used to measure die temperature and voltage on the hw_device. The Sysmon monitors the physical environment via on-chip temperature and supply sensors. The ADC can access up to 17 external analog input channels.

Data for the system monitor is stored in dedicated registers, called status and control registers, accessible through the get_hw_sysmon_reg and set_hw_sysmon_reg commands. Refer to the Register Interface in *UltraScale Architecture System Monitor User Guide (UG580),* or *7 Series FPGAs and Zynq-7000 SoC XADC Dual 12-Bit 1 MSPS Analog-to-Digital Converter User Guide (UG480)* for more information on the addresses of specific system monitor registers.

Although the `set_hw_sysmon_reg` command lets you directly write the specified hex data value into the registers of a system monitor, the recommended procedure is to update the values of properties on the hw_sysmon object using the `set_property` command, and then write the property values to the hw_sysmon object using the `commit_hw_sysmon` command.

The `set_hw_sysmon_reg` command writes the specified hex value to the hw_sysmon_reg object on the hw_sysmon object at the specified address but returns nothing, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_sysmon> - (Required) Specify the hw_sysmon object to set the registers of. The hw_sysmon object must be specified as an object returned by the `get_hw_sysmon` command.

<hexaddress> - (Required) Specify the hex address of the status register on the system monitor to set the value of.

<hexdata> - (Required) Specify the data, as a hex value, to populate into the register defined by the hex address.

Example

The following example sets the specified hex data value into the register at the hex address of the system monitor on the current hardware device:

```
set_hw_sysmon_reg [current_hw_device] 00 9D28
```

See Also

- commit_hw_sysmon
- connect_hw_server
- current_hw_device
- get_hw_sysmons
- get_hw_sysmon_reg
- open_hw_target
- refresh_hw_sysmon

- set_property

set_input_delay

Set input delay on ports

Syntax

```
set_input_delay [-clock <args>] [-reference_pin <args>] [-clock_fall]
    [-rise] [-fall] [-max] [-min] [-add_delay] [-network_latency_included]
    [-source_latency_included] [-quiet] [-verbose] <delay> <objects>
```

Usage

Name	Description
[-clock]	Relative clock
[-reference_pin]	Relative pin or port
[-clock_fall]	Delay is relative to falling edge of clock
[-rise]	Specifies rising delay
[-fall]	Specifies falling delay
[-max]	Specifies maximum delay
[-min]	Specifies minimum delay
[-add_delay]	Don't remove existing input delay
[-network_latency_included]	Specifies network latency of clock already included
[-source_latency_included]	Specifies source latency of clock already included
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<delay>	Delay value
<objects>	List of ports

Categories

[SDC](#), [XDC](#)

Description

> 💡 **TIP:** *The XDC > Timing Constraints language templates and the Timing Constraints Wizard in the Vivado IDE offer timing diagrams and additional details around defining specific timing constraints. You can refer to these sources for additional information.*

Specifies the external system-level path delay on a primary input port relative to a clock edge at the interface of the design. The input delay value is specified in nanoseconds (ns), and can be positive or negative, depending on the clock and data relative phase at the interface of the device.

Send Feedback

To accurately model the system-level timing of your Xilinx FPGA design, you must assign timing delays for objects external to the FPGA onto the primary input or output ports in your design. These delays are defined by the `set_input_delay` and `set_output_delay` commands.

> ⭐ **IMPORTANT!** *If the input port also has a `set_max_delay` constraint assigned, the specified input delay value is considered part of the max_delay computation. That is, the input delay consumes a portion of the max delay on the timing path that includes the input port.*

This command returns nothing if successful, or returns an error if it fails.

Arguments

`-clock <arg>` - (Optional) Indicates that the input delay is relative to the specified clock. By default the rising edge is used. However the `-clock_fall` argument can be used to indicate that the falling edge should be used instead.

`-reference_pin <arg>` - (Optional) Specifies that the delay is relative to the active edge of a clock appearing on the specified pin or port rather than a clock.

`-clock_fall` - (Optional) Specifies that the delay is relative to a falling edge of the clock rather than rising edge.

`-rise` - (Optional) Specifies the input delay applies to rising transitions on the specified ports. The default is to apply the delay for both rising and falling transitions.

`-fall` - (Optional) Specifies the input delay applied to falling transitions on the specified ports. The default is to apply the delay for both rising and falling transitions.

`-max` - (Optional) Indicates the input delay specified is only used when calculating the maximum (longest) path delays.

`-min` - (Optional) Indicates the input delay specified is only used when calculating the minimum (shortest) path delays.

`-add_delay` - (Optional) Add the specified delay constraint to the port, to coexist with any other `set_input_delay` constraints already defined on the port. The default behavior is to replace the existing delays.

`-network_latency_included` - (Optional) Indicates that the clock network latency of the reference clock is included in the delay value. The Vivado timing engine considers the clock edge reaching the capture flop after the clock latencies unless the specified input or output delay value includes the source latency or network latency.

`-source_latency_included` - (Optional) Indicates that the source latency of the relative clock is included in the specified delay value.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<delay> - (Required) The input delay specified as nanoseconds (ns) to apply to the specified ports. Valid values are floating point numbers, with a default value of 0.

`<objects>` - (Required) The list of ports to which the delay value will be assigned.

Examples

The following example specifies the input delay on port DIN. The input delay is 3 and is relative to the rising edge of clock clk1:

```
set_input_delay -clock clk1 3 DIN
```

The following example specifies the input delay on port DIN. The input delay is 2 and is relative to the falling edge of the clock clk1:

```
set_input_delay -clock_fall -clock clk1 2 DIN
```

The following example specifies the input delay on port reset. The input delay is 2 and is relative to the rising edge of the clock that appears on the pin wbClk_IBUF_BUFG_inst/O, originating from the clock wbClk:

```
set_input_delay -clock wbClk 2 -reference_pin \
    [get_pin wbClk_IBUF_BUFG_inst/O] reset
```

This example creates a clock named clk_ddr, and defines input delay constraints from data launched by both rising and falling edges of the clock outside the device to the data input of the internal flip-flop that is sensitive to both rising and falling clock edges:

```
create_clock -name clk_ddr -period 6 [get_ports DDR_CLK_IN]
set_input_delay -clock clk_ddr -max 2.1 [get_ports DDR_IN]
set_input_delay -clock clk_ddr -max 1.9 [get_ports DDR_IN] -clock_fall -
add_delay
set_input_delay -clock clk_ddr -min 0.9 [get_ports DDR_IN]
set_input_delay -clock clk_ddr -min 1.1 [get_ports DDR_IN] -clock_fall -
add_delay
```

Note: The use of the `-add_delay` option allows the new min and max delay constraints to exist alongside the first delays on the same port.

Send Feedback

The following example specifies the input delay on all non clock input ports of the design. Although all_inputs returns all ports of the design, including clock ports, set_input_delay will skip setting input delays on the clock ports. The input delay is 1 relative to the rising edge of the clock wbClk:

```
set_input_delay -clock wbClk 1 [all_inputs]
```

The following example sets an input delay of 4 relative to the rising edge of the clock wbClk on the ports reset and wbDataForInput:

```
set_input_delay -clock wbClk 4 [list reset wbDataForInput]
```

See Also

- all_clocks
- all_inputs
- check_timing
- create_clock
- get_ports
- report_timing
- set_max_delay
- set_output_delay

Send Feedback

set_input_jitter

Set input jitter for a clock object

Syntax

```
set_input_jitter [-quiet] [-verbose] <clock> <input_jitter>
```

Returns

Clock

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<clock>	Clock
<input_jitter>	Input jitter: Value >= 0

Categories

XDC

Description

Use the `set_input_jitter` command to specify additional jitter for a specific primary clock.

Input jitter is the difference between successive clock edges due to variation from the ideal arrival times. This command sets the input jitter in nanoseconds (ns) for a specified primary clock, defined with the `create_clock` command. Because the command accepts a single clock, the jitter for each primary clock must be set individually.

You can only use the `set_input_jitter` command to specify input jitter on primary clocks. You cannot use the command to set input jitter on generated or auto derived clocks. Input jitter is propagated to generated clocks from the master clock, except for MMCM and PLL.

The input jitter is used in the calculation of discrete jitter, which is the amount of jitter introduced by hardware primitives such as MMCM or PLL. Discrete jitter is a feature of clocks generated by the MMCM. See `set_clock_uncertainty`.

The `set_input_jitter` command is ignored during synthesis.

> **TIP:** *INPUT_JITTER is a property of primary clocks that can be returned by the* `get_property` *or* `report_property` *commands.*

This command returns nothing if successful, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<clock> - (Required) The clock name of a primary clock, defined with the `create_clock` command.

<input_jitter> - (Required) The input jitter for the specified clock object (value >= 0). The value is specified as nanoseconds (ns).

Examples

The following example sets an input jitter value of 0.3 ns on two clocks, sysClk and procClk. Although the jitter values are the same, you must use two `set_input_jitter` commands since the command only takes one clock as an argument:

```
set_input_jitter sysClk 0.3
set_input_jitter procClk 0.3
```

The following example defines a primary clock, sysClk, and a generated clock, sysClkDiv2, that is a divide by two version of the primary clock. An input jitter of 0.15 ns is specified on the primary clock. The input jitter is automatically propagated to the generated clock:

```
create_clock -period 10 -name sysClk [get_ports sysClk]
create_generated_clock  -name sysClkDiv2 -source [get_ports sysClk] \
    -divide_by 2 [get_pins clkgen/sysClkDiv/Q]
set_input_jitter sysClk 0.15
```

Note: In this example sysClkDiv2 is generated by a divider implemented with flip-flops, so the input jitter is propagated from the primary clock.

See Also

- all_clocks
- check_timing
- create_clock
- create_generated_clock

Send Feedback

- report_clocks
- report_timing
- set_clock_uncertainty
- set_clock_latency
- set_system_jitter

set_load

Set capacitance on output ports and output pins

Syntax

```
set_load [-rise] [-fall] [-max] [-min] [-quiet] [-verbose] <capacitance>
    <objects>
```

Usage

Name	Description
[-rise]	Specify the rise capacitance value (for ports only)
[-fall]	Specify the fall capacitance value (for ports only)
[-max]	Specify the maximum capacitance value
[-min]	Specify the minimum capacitance value
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<capacitance>	Capacitance value
<objects>	List of output ports and output pins

Categories

SDC, XDC

Description

Sets the load capacitance on output ports to the specified value. The load capacitance is used during power analysis when running the `report_power` command, but is not used during timing analysis.

> 💡 **TIP:** *The default unit of capacitance is picofarads (pF), but can be changed using the* `set_units` *command.*

This command operates silently and does not return direct feedback of its operation.

Arguments

`-max` - (Optional) Provided for SDC compatibility. Ignored by the Vivado tools.

`-min` - (Optional) Provided for SDC compatibility. Ignored by the Vivado tools.

`-rise` - (Optional) Provided for SDC compatibility. Ignored by the Vivado tools.

`-fall` - (Optional) Provided for SDC compatibility. Ignored by the Vivado tools.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<capacitance> - (Required) The value of the load capacitance. The value is specified as a floating point value >= 0. The default is 0.

`<objects>` - (Required) A list of output port objects to assign the capacitance load to. All outputs in the design may be obtained using the `all_outputs` command.

Examples

The following example sets the specified load capacitance value for all ports:

```
set_load 5.5 [all_outputs]
```

The following example sets the rising and falling edge load capacitance for the specified output ports:

```
set_load -rise -fall 8 [get_ports wbOutput*]
```

See Also

- all_outputs
- get_ports
- report_power

set_logic_dc

Sets logic dc for input ports and input pins

Syntax

```
set_logic_dc [-quiet] [-verbose] <objects>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<objects>	List of input ports and input pins

Categories

SDC, XDC

Description

Sets the specified input ports or input pins to a logic value of 'X', as unknown or don't care. This command is NOT supported in Synthesis.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<objects>` - (Required) A list of the input ports and pins to be affected.

Examples

The following example sets the specified port to 'X':

```
set_logic_dc [get_ports reset]
```

Send Feedback

See Also

- all_inputs
- get_pins
- get_ports
- set_logic_one
- set_logic_unconnected
- set_logic_zero

set_logic_one

Sets logic one for input ports and input pins

Syntax

```
set_logic_one [-quiet] [-verbose] <objects>
```

Usage

Name	Description
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<objects>`	List of input ports and input pins

Categories

SDC, XDC

Description

Sets the specified input ports or input pins to a logic one. This command is NOT supported in Synthesis.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<objects>` - (Required) A list of the input ports and pins to be affected.

Examples

The following example sets the specified input port to a logic one:

```
set_logic_one [get_ports reset]
```

The following example sets the input ports reset and wbDataForInput to a logic one:

```
set_logic_one [list [get_ports reset] [get_ports wbDataForInput]]
```

The following example sets the input pin I on instance reset_IBUF to a logic one:

```
set_logic_one [get_pins reset_IBUF_inst/I]
```

See Also

- all_inputs
- get_pins
- get_ports
- set_logic_dc
- set_logic_unconnected
- set_logic_zero

Send Feedback

set_logic_unconnected

Sets logic unconnected for output ports and output pins

Syntax

```
set_logic_unconnected [-quiet] [-verbose] <objects>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<objects>	List of output ports and output pins

Categories

XDC

Description

Defines the specified output ports or pins as unconnected.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<objects>` - (Required) A list of the output ports and pins to be affected.

Examples

The following example sets the specified port to unconnected:

```
set_logic_unconnected [get_ports OUT1]
```

See Also

- set_logic_dc
- set_logic_one
- set_logic_zero

set_logic_zero

Sets logic zero for input ports and input pins

Syntax

```
set_logic_zero [-quiet] [-verbose] <objects>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<objects>	List of input ports and input pins

Categories

SDC, XDC

Description

Sets the specified input ports and input pins to a logic zero. This command is NOT supported in Synthesis.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<objects>` - (Required) A list of the input ports and pins to be affected.

Examples

The following example sets the specified port to logic state 0:

```
set_logic_zero [get_ports reset]
```

Send Feedback

See Also

- all_inputs
- get_pins
- get_ports
- set_logic_one
- set_logic_unconnected

set_max_delay

Specify maximum delay for timing paths

Syntax

```
set_max_delay [-rise] [-fall] [-reset_path] [-from <args>]
    [-rise_from <args>] [-fall_from <args>] [-to <args>] [-rise_to <args>]
    [-fall_to <args>] [-through <args>] [-rise_through <args>]
    [-fall_through <args>] [-datapath_only] [-quiet] [-verbose] <delay>
```

Usage

Name	Description
[-rise]	Delay value applies to rising path delays
[-fall]	Delay value applies to falling path delays
[-reset_path]	Reset this path before setting max delay
[-from]	List of path startpoints or clocks
[-rise_from]	Apply to paths rising from the list of startpoints or clocks
[-fall_from]	Apply to paths falling from the list of startpoints or clocks
[-to]	List of path endpoints or clocks
[-rise_to]	Apply to paths with rise transition at the list of endpoints or clocks
[-fall_to]	Apply to paths with fall transition at the list of endpoints or clocks
[-through]	List of through pins, cells or nets
[-rise_through]	Apply to paths rising through pins, cells or nets
[-fall_through]	Apply to paths falling through pins, cells or nets
[-datapath_only]	Remove clock skew and jitter from calculation
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<delay>	Delay value

Categories

[SDC](#), [XDC](#)

Description

> 💡 **TIP:** *The XDC > Timing Constraints language templates and the Timing Constraints Wizard in the Vivado IDE offer timing diagrams and additional details around defining specific timing constraints. You can refer to these sources for additional information.*

Sets the maximum delay allowed on a timing path, specified in nanoseconds (ns). The specified delay value is assigned to both the rising and falling edges of the defined timing paths unless the -rise or -fall arguments are specified.

Send Feedback

The maximum rising and falling delay cannot be less than the minimum rising and falling delay on the same path, as defined by the `set_min_delay` command. If this happens, the first assigned constraint is removed from the timing path as a conflict, and the delay value specified by the removed constraint is set to 0.

The delay value must be assigned to a timing path as defined by at least one `-from`, `-through`, or `-to` argument. A general path delay such as `-to` endpoint will be over written by a more specific path definition such as `-from/-to`, or `-from/-through/-to` path definition.

> **IMPORTANT!** *When assigned to a primary input or output port, any system-level delay consumes a portion of the max delay on the timing path that includes the input or output port. That is, the delay specified by* `set_input_delay` *or* `set_output_delay` *is considered part of the maximum delay.*

This command returns nothing if successful, or returns an error if it fails.

Arguments

`-rise` - (Optional) Apply the delay value to the rising edge of the timing path.

`-fall` - (Optional) Apply the delay value to the falling edge of the timing path.

Note: If neither `-rise` nor `-fall` is specified, the delay is applied as both rising and falling edge path delay.

`-reset_path` - (Optional) Indicates that existing rising or falling edge false path or max delays should be cleared before applying the new specified path delay. If only `-to` is specified all paths leading to the specified endpoints are cleared. If only `-from` is specified, all paths leading from the specified start points are cleared. When `-from/-to` or `-from/-through/-to` are specified, the defined paths are reset.

`-from <value>` - (Optional) A list of path start points or clocks. A valid startpoint is a primary input or inout port, or the clock pin of a sequential element. If a clock is specified then all the primary input and inout ports related to that clock as well as all the clock pin of the registers connected to that clock are used as startpoints.

`-rise_from <element_name>` - (Optional) The max delay applied to paths rising from the list of origins or clocks.

`-fall_from <element_name>` - (Optional) The max delay applied to paths falling from the list of origins or clocks.

`-to <element_name>` - (Optional) A list of path endpoints or clocks. A valid endpoint is a primary output or inout port, or the data pin of a sequential element. If a clock is specified then all the primary output and inout ports related to that clock as well as all the data pins of the registers connected to that clock are used as endpoints.

`-rise_to <element_name>` - (Optional) The max delay applied to paths with rise transition at the list of endpoints or clocks.

`-fall_to` <element_name> - (Optional) The max delay applied to paths with fall transition at the list of endpoints or clocks.

`-through` <element_name> - (Optional) A list of through pins, cells, or nets.

`-rise_through` <element_name> - (Optional) The max delay applied to paths rising through pins, cells or nets.

`-fall_through` <element_name> - (Optional) The max delay applied to paths falling through pins, cells or nets.

`-datapath_only` - (Optional) Exclude clock skew and jitter from the delay calculation for the specified path. This option is used to constrain the delay between sequential elements that have different clocks, where you do not want to consider clock skew and jitter in the delay calculation. Only the Clock-to-Q delay of the first flop, the wire delay between the flops, and the setup time of the second flop should be considered.

> ⭐ **IMPORTANT!** *The specification of the data path for* `-datapath_only` *must use the* `-from` *option to define the startpoint for the path. The hold check for the specified path is automatically defined as a false path when using the* `-datapath_only` *option.*

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<delay> - (Required) Specifies the maximum delay value specified in nanoseconds. The <delay> is specified in nanoseconds (ns) as a positive or negative floating point number, with a default value of 0.

Examples

The following example defines a maximum delay of 60 ns between all the input and output ports (feedthrough paths):

```
set_max_delay 60 -from [all_inputs] -to [all_outputs]
```

The following example clears the existing max delay and specifies a new > maximum delay for paths to endpoints clocked by the specified clock:

```
set_max_delay -reset_path 50 -to [get_clocks spi_clk]
```

Send Feedback

The set_max_delay command is often used with -datapath_only to constrain asynchronous clock domains crossing when a simple synchronizer is used on the destination clock domain. In the following example, two flops (FF1 and FF2) are clocked by different clocks, and FF1/Q connects directly to FF2/D through net1. To limit the delay on this connection to 4.0 ns use the following constraints:

```
set_max_delay -from FF1/C -to FF2/D -datapath_only 4.0
```

See Also

- get_clocks
- get_nets
- get_ports
- report_timing
- set_input_delay
- set_min_delay
- set_output_delay
- set_units

set_max_time_borrow

Limit time borrowing for latches

Syntax

```
set_max_time_borrow [-quiet] [-verbose] <delay> <objects>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<delay>	Delay value: Value >= 0
<objects>	List of clocks, cells, data pins or clock pins

Categories

SDC, XDC

Description

Sets the maximum amount of time in nanoseconds that can be borrowed between nets when analyzing the timing on latches.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the set_msg_config command.

<delay> - (Required) The delay that should be applied to the specified objects. The <delay> is specified in nanoseconds (ns) as a floating point number >= 0, with a default value of 0.

<objects> - (Required) A list of clocks, cells, data pins, or clock pins to which the limit should be applied.

Examples

The following example specifies that the latches attached to "all clocks" will be allowed 0 time units of borrowing. Effectively, this disables time borrowing throughout the entire design.

```
set_max_time_borrow 0.0 [all_clocks]
```

The following example specifies that nets in the top level of hierarchy are allowed 20 time units of time borrowing:

```
set_max_time_borrow 20 {top/*}
```

See Also

- all_clocks
- get_clocks
- get_nets

set_min_delay

Specify minimum delay for timing paths

Syntax

```
set_min_delay [-rise] [-fall] [-reset_path] [-from <args>]
    [-rise_from <args>] [-fall_from <args>] [-to <args>] [-rise_to <args>]
    [-fall_to <args>] [-through <args>] [-rise_through <args>]
    [-fall_through <args>] [-quiet] [-verbose] <delay>
```

Usage

Name	Description
[-rise]	Delay value applies to rising path delays
[-fall]	Delay value applies to falling path delays
[-reset_path]	Reset this path before setting min delay
[-from]	List of path startpoints or clocks
[-rise_from]	Apply to paths rising from the list of startpoints or clocks
[-fall_from]	Apply to paths falling from the list of startpoints or clocks
[-to]	List of path endpoints or clocks
[-rise_to]	Apply to paths with rise transition at the list of endpoints or clocks
[-fall_to]	Apply to paths with fall transition at the list of endpoints or clocks
[-through]	List of through pins, cells or nets
[-rise_through]	Apply to paths rising through pins, cells or nets
[-fall_through]	Apply to paths falling through pins, cells or nets
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<delay>	Delay value

Categories

[SDC](SDC), [XDC](XDC)

Description

Sets the minimum delay allowed on a timing path, specified in nanoseconds (ns). The specified delay value is assigned to both the rising and falling edges of the defined timing paths unless the -rise or -fall arguments are specified.

> ⭐ **IMPORTANT!** *The minimum rising and falling delay cannot be greater than the maximum rising and falling delay on the same path. If this happens, the first assigned delay value is removed from the timing path and reset to 0.*

The delay value must be assigned to a timing path as defined by at least one `-from`, `-through`, or `-to` argument. A general path delay such as `-to` endpoint will be over written by a more specific path definition such as `-from/-to`, or `-from/-through/-to` path definition.

This command operates silently and does not return direct feedback of its operation.

Arguments

`-rise` - (Optional) Apply the delay value to the rising edge of the timing path.

`-fall` - (Optional) Apply the delay value to the falling edge of the timing path.

`-reset_path` - (Optional) Clear existing rising or falling edge min delays before applying the new specified path delay. If only `-to` is specified all paths leading to the specified endpoints are cleared. If only `-from` is specified, all paths leading from the specified starting points are cleared. When `-from/-to` or `-from/-through/-to` are specified, the defined paths are reset.

`-from <objects>` - (Optional) The starting points of the timing paths that will be assigned the specified delay. A valid startpoint is a primary input or inout port, or the clock pin of a sequential element. If a clock is specified then all the primary input and inout ports related to that clock, as well as all the clock pins of the registers connected to that clock are used as startpoints.

`-rise_from <objects>` - (Optional) The starting points of the timing path that will have the specified delay assigned to its rising edge.

`-fall_from <objects>` - (Optional) The starting points of the timing path that will have the specified delay assigned to its falling edge.

`-to <objects>` - (Optional) The destination objects for the path that will be affected by the minimum delay. A valid endpoint is a primary output or inout port, or the data pin of a sequential element. If a clock is specified then all the primary output and inout ports related to that clock, as well as all the data pins of the registers connected to that clock are used as endpoints.

`-rise_to <objects>` - (Optional) The destination objects for the rising-edge path that will be affected by the minimum delay.

`-fall_to <objects>` - (Optional) The destination objects for the falling-edge path that will be affected by the minimum delay.

`-through <objects>` - (Optional) A list of pins, cell, or nets through which the path affected by the minimum delay travels.

`-rise_through <objects>` - (Optional) A list of pins, cell, or nets through which the rising-edge path affected by the minimum delay travels.

`-fall_through <objects>` - (Optional) A list of pins, cell, or nets though which the falling-edge path affected by the minimum delay travels.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<delay> - (Required) Specifies the minimum delay value in nanoseconds. The <delay> is specified in nanoseconds (ns) as a positive or negative floating point number, with a default value of 0.

Examples

The following example specifies a minimum delay of 20ns between the primary input and output ports (combinational/feedthrough paths):

```
set_min_delay 20 -from [all_inputs] -to [all_outputs]
```

The following example defines a minimum delay of 20ns for timing paths with endpoints at all primary output ports:

```
set_min_delay 20 -to [get_ports -filter {DIRECTION == out}]
```

See Also

- get_clocks
- get_nets
- get_ports
- report_timing
- set_max_delay

set_msg_config

Configure how the Vivado tool will display and manage specific messages, based on message ID, string, or severity.

Syntax

```
set_msg_config [-id <arg>] [-string <args>] [-severity <arg>]
    [-limit <arg>] [-new_severity <arg>] [-suppress] [-regexp] [-quiet]
    [-verbose]
```

Usage

Name	Description
[-id]	A qualifier, apply the selected operation only to messages that match given message id. Example: '-id {Common 17-35}'. Default: match any id
[-string]	A qualifier, apply the selected operation only to messages that contain the given list of strings. Default: none
[-severity]	A qualifier, apply the selected operation only to messages at the given severity level. Example: '-severity INFO' Default: match any severity
[-limit]	for the messages that match the qualifiers, limit the number of messages displayed to the given integer value. Can only be used in conjunction with one of -id or -severity.
[-new_severity]	for the messages that match the qualifiers, change the severity to the given value for the current project
[-suppress]	for the messages that match the qualifiers, suppress (do not display) any messages for the current project
[-regexp]	The values used for -string are full regular expressions
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Report](#)

Description

This command lets you configure the messages returned by the Vivado tool in the current project. Use this command to change the severity of messages, to limit the number of times a message is reported, or to suppress the message altogether. However, you can only perform one of these actions at one time with `set_msg_config`:

- Customize the severity of messages returned by the tool to specific levels appropriate to your usage. For instance, set the severity of a specified message ID from one type, such as WARNING, to another type, such as ERROR.

Send Feedback

> ⭐ **IMPORTANT!** *You cannot downgrade a Vivado Design System ERROR message to make it less than an error.*

- Define the number of messages that will be returned by the tool during a design session, or single invocation. You can specify the limit of a specific message ID, or the limit for a specific severity of messages.

> 💡 **TIP:** *The default message limit for all message IDs is set to 100, and is defined by the parameter* `messaging.defaultLimit`*. This is the limit applied to each separate message returned by the tool. You can report the current value of this parameter with the* `get_param` *command, and change it as needed using the* `set_param` *command.*

- Suppress a specific message ID from being reported by the tool at all. You can enable messages that were previously suppressed using the `reset_msg_config` command.

- An error is returned if more than one action is attempted in a single `set_msg_config` command.

Message qualifiers of string, ID, and severity are used to determine which messages are configured by the `set_msg_config` command. You must supply at least one message qualifier to identify a message or group of messages to apply the command to. Multiple qualifiers have an AND relationship; the configuration rule will be applied only to messages matching all qualifiers.

> 💡 **TIP:** `set_msg_config` *does not support the use of wildcards in message qualifiers.*

Message configuration rules are project specific, and are persistent with the project when the project is closed and reopened.

> ⭐ **IMPORTANT!** *Message configuration rules apply to the current project and are passed automatically to subordinate processes, such as synthesis and implementation runs. Do not use* `set_msg_config` *in pre and post Tcl scripts.*

Use the `get_msg_config` command to report the current configuration of a specific message, or the configuration rules defined in the current project. Restore messages to their default configurations using the `reset_msg_config` command.

The `set_msg_config` command is not supported by `report_cdc` as that command does not generate messages through the message manager.

This command returns nothing if successful, or returns an error if it fails.

Arguments

`-id <arg>` - (Optional) Specify a message ID pattern to find message IDs that match the specified argument. The specified <arg> is used as a search pattern. All message IDs that match the specified pattern will be affected by the `set_msg_config` command. Every message delivered by the tool has a unique global message ID that consists of an application sub-system code and a message identifier. This results in a message ID that looks like the following:

```
[Common 17-54]
[Netlist 29-28]
[Synth 8-32]
[Synth 8-3295]
```

> 💡 **TIP:** *To match a specific message ID, make the search pattern specific to the ID. For instance, in the following commands, the first applies to both "Synth 8-32" and "Synth 8-3295", while the second command applies only to "Synth 8-32":*
>
> ```
> set_msg_config -id "Synth 8-32" -new_severity "CRITICAL WARNING"
> set_msg_config -id {[Synth 8-32]} -new_severity "CRITICAL WARNING"
> ```

`-string <args>` - (Optional) Apply the selected operation only to messages that contain the given list of strings. Strings must be enclosed in braces, and multiple strings can be specified separated by spaces:

```
{{Vivado} {Synthesis}}
```

Note: Strings are case sensitive.

`-severity` - (Optional) The severity of the message. There are five message severities:

- `ERROR` - An ERROR condition implies an issue has been encountered which will render design results unusable and cannot be resolved without user intervention.

- `{CRITICAL WARNING}` - A CRITICAL WARNING message indicates that certain input/constraints will either not be applied or are outside the best practices for a FPGA family. User action is strongly recommended.

  Note: Since this is a two word value, it must be enclosed in "" or {}.

- `WARNING` - A WARNING message indicates that design results may be sub-optimal because constraints or specifications may not be applied as intended. User action may be taken or may be reserved.

- `INFO` - An INFO message is the same as a STATUS message, but includes a severity and message ID tag. An INFO message includes a message ID to allow further investigation through answer records if needed.

- `STATUS` - A STATUS message communicates general status of the process and feedback to the user regarding design processing. A STATUS message does not include a message ID.

Note: Because STATUS messages do not have message IDs, you cannot change the severity level of a STATUS message.

`-limit <arg>` - (Optional) Limit the display of the selected messages by the limit value specified as an integer >= 1. You can restore the message limit to the `messaging.defaultLimit` by specifying a count of -1.

`-new_severity <arg>` - (Optional) For the messages that match the qualifier, specify a new message severity. Valid values are defined above under the `-severity` option.

> ⭐ **IMPORTANT!** *Using `-new_severity` with `-id` or `-string` may appear to let you downgrade an ERROR message when the command is run. However, the ERROR message is not downgraded. This will be correctly reported by the Vivado tool the next time the error is encountered. See the Examples section below for more information.*

`-suppress` - (Optional) Suppress the specified messages from further reporting.

> ⚠ **CAUTION!** *Suppressing all messages of a specified severity, such as WARNING, can suppress implementation, DRC, and clock domain crossing (CDC) messages related to potential problems in your design.*

`-regexp` - (Optional) Can be used with `-string` to specify the string values as regular expressions. Regular expressions search strings are anchored to the start of the search string. You can add ".*" to the beginning and end of a string to widen the search to include a sub-string. See http://perldoc.perl.org/perlre.html for more information on regular expression syntax.

Note: The Tcl built-in command `regexp` is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example elevates a common INFO message to a Critical Warning:

```
set_msg_config -id {[Common 17-81]} -new_severity "CRITICAL WARNING"
```

⭐ **IMPORTANT!** *In the following example the "Common 17-69" message is an ERROR message, and cannot be downgraded from an ERROR. The command in this example appears to work when run from the Tcl console, however it will not result in any change.*

```
set_msg_config -id {[Common 17-69]} -new_severity WARNING
```

When the "Common 17-69" message is next thrown by the Vivado tool, a warning message is returned stating that an error cannot be downgraded, and the message is thrown as an ERROR:

```
WARNING: [Common 17-239] ERROR Messages are prohibited to be downgraded.
    Message 'Common 17-69' is not downgraded.
ERROR: [Common 17-69] Command failed: report_design_analysis
    -critical_paths can be run only after synthesis has successfully
completed.
```

The following example results in warning messages with message ID "17-35", and containing "clk" in the message, being redefined as Error messages:

```
set_msg_config -severity warning -string "clk" -id "17-35" \
    -new_severity error
```

This example changes the severity of messages with the specified message ID, gets the current message configuration rules, and then shows two different command forms to reset the specific rule and restore the message:

```
set_msg_config -id "Common 17-361" -severity INFO -new_severity WARNING
get_msg_config -rules
--------------------
Message control rules currently in effect are:
Rule Name   Rule                                              Current
Message Count
1  set_msg_config -ruleid {1} -id {Common 17-361} -severity {INFO} -
new_severity {WARNING} 0
--------------------
reset_msg_config -id "Common 17-361" -default_severity
reset_msg_config -ruleid {1}
```

💡 **TIP:** *In the preceding example, only one of the `reset_msg_config` commands is needed to reset the message.*

This example shows the use of a parameter to change the default message limit, and then defines a new limit for the specified message id:

```
get_param messaging.defaultLimit
    100
set_param messaging.defaultLimit 1000
set_msg_config -id {[Common 17-81]} -limit 1500
```

See Also

- get_msg_config
- get_param

Send Feedback

- reset_msg_config
- set_param

Send Feedback

set_multicycle_path

Define multicycle path

Syntax

```
set_multicycle_path [-setup] [-hold] [-rise] [-fall] [-start] [-end]
    [-reset_path] [-from <args>] [-rise_from <args>] [-fall_from <args>]
    [-to <args>] [-rise_to <args>] [-fall_to <args>] [-through <args>]
    [-rise_through <args>] [-fall_through <args>] [-quiet] [-verbose]
    <path_multiplier>
```

Usage

Name	Description
[-setup]	Only setup multiplier is set
[-hold]	Only hold multiplier is set
[-rise]	Multiplier valid for rising delays on path endpoint
[-fall]	Multiplier valid for falling delays on path endpoint
[-start]	Multiplier measured against path startpoint
[-end]	Multiplier measured against path endpoint
[-reset_path]	Reset this path before setting multicycle
[-from]	List of path startpoints or clocks
[-rise_from]	Apply to paths rising from the list of startpoints or clocks
[-fall_from]	Apply to paths falling from the list of startpoints or clocks
[-to]	List of path endpoints or clocks
[-rise_to]	Apply to paths with rise transition at the list of endpoints or clocks
[-fall_to]	Apply to paths with fall transition at the list of endpoints or clocks
[-through]	List of through pins, cells or nets
[-rise_through]	Apply to paths rising through pins, cells or nets
[-fall_through]	Apply to paths falling through pins, cells or nets
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<path_multiplier>	Number of cycles

Categories

[SDC](), [XDC]()

Send Feedback

Description

By default, the Vivado timing engine performs a single-cycle analysis, in which the setup check is performed at the destination on the capture edge, one clock cycle after the edge of the source clock. However, this may not be appropriate for certain timing paths. The most common example is a logic path that requires more than one clock cycle for the data to stabilize at the endpoint.

The `set_multicycle_path` command lets you choose a path multiplier, N, to establish a timing path that takes N clock cycles from the start clock edge to the capture clock edge. The path multiplier defines the total number of clock cycles required for propagation of a signal from its origin to destination when that propagation is longer than a single clock cycle. For more information on the use of this command, refer to the *Vivado Design Suite User Guide: Using Constraints (UG903)*.

The `set_multicycle_path` command is used to specify path multipliers for setup and hold analysis, for rising and/or falling edges, with respect to the source clock or the destination clock. This command includes three elements:

- The specification of the setup and hold analysis affected by the multicycle path.

- The definition of the timing paths to which the multicycle path applies.

- The path multiplier defining the number of clock cycles to apply to the timing analysis.

By default the path multiplier applies to both the setup and hold analysis. The hold analysis is derived from the setup analysis, so it is moved along with the setup analysis. If the path multiplier moves the setup check N clock cycles, it moves the hold check N-1 clock cycles. However, this often results in hold timing failures.

You can use a second `set_multicycle_path` command with the `-hold` option to restore the hold analysis to its original location. When the `-hold` option is specified the <path_multiplier> acts on the hold relationship to restore the hold check to its original position. For instance, the following command sequence extends the setup check for 3 clock cycles, and consequently extends the hold check by two clock cycles (N-1). The second command restores the hold check to its original position:

```
set_multicycle_path 3 -from {usbEngine1/u4/csr_reg[26]/C} \
-to {usbEngine1/u1/u2/sizd_c_reg[12]/D}
set_multicycle_path 2 -from {usbEngine1/u4/csr_reg[26]/C} \
-to {usbEngine1/u1/u2/sizd_c_reg[12]/D} -hold
```

By default, the setup path multiplier is applied with respect to the destination clock, and the hold path multiplier is applied with respect to the source clock. Use the `-start` or `-end` options to change the default setup or hold analysis with respect to the source or destination clocks.

This command operates silently when successful, or returns an error if the command fails.

Send Feedback

Arguments

`-setup` - (Optional) Apply the path multiplier to the setup check, which also affects the hold check. This is also the default behavior of the `set_multicycle_path` command when neither `-setup` nor `-hold` are specified.

`-hold` - (Optional) Apply the path multiplier only to the hold check, to change the hold relationship by the specified number of clock cycles.

Note: When neither `-setup` nor `-hold` is used, or when only `-setup` is specified, the <path_multiplier> applies to both setup and hold checks.

`-rise` - (Optional) Apply the multiplier specifically to rising edge delays on the path endpoint.

`-fall` - (Optional) Apply the multiplier specifically to falling edge delays on the path endpoint.

Note: If neither `-rise` or `-fall` is specified, the multiplier is applied to both the rising and falling edge delays.

`-start` - (Optional) By default, the setup path multiplier is defined with respect to the destination clock (`-end`). To modify the setup requirement with respect to the source clock, the `-start` option must be used.

`-end` - (Optional) By default, the hold path multiplier is defined with respect to the source clock. To modify the hold requirement with respect to the destination clock, the `-end` option must be used.

Note: The `-start/-end` options have no effect when applying a multicycle path constraint on paths clocked by the same clock, or clocked by two clocks having the same waveform, or with no phase shift.

`-reset_path` - (Optional) Reset the specified path before applying the multicycle path multiplier.

`-from` <args> - (Optional) A list of start points on the path that will be affected by the path multiplier.

`-rise_from` <args> - (Optional) A list of the start points on the rising-edge path that will be affected by the multicycle path multiplier.

`-fall_from` <args> - (Optional) A list of the start points on the falling-edge path that will be affected by the multicycle path multiplier.

`-to` <args> - (Optional) A list of the end points on the path that will be affected by the multicycle path multiplier.

`-rise_to` <args> - (Optional) A list of the end points on the rising-edge path that will be affected by the multicycle path multiplier.

Send Feedback

-`fall_to` <args> - (Optional) A list of the end points on the falling-edge path that will be affected by the multicycle path multiplier.

-`through` <args> - (Optional) A list of pins, cell, or nets through which the path affected by the multicycle path multiplier travels.

-`rise_through` <args> - (Optional) A list of pins, cell, or nets through which the rising-edge path affected by the multicycle path multiplier travels.

-`fall_through` <args> - (Optional) Specifies the list of pins, cell, or nets through which the falling-edge path affected by the multicycle path multiplier travels.

> **IMPORTANT!** *Although* `-to`, `-through`, *and* `-from` *(in their various forms) are all optional arguments, at least one* `-from`, `-to`, *or* `-through` *argument must be specified to define a timing path for the* `set_multicycle_path` *constraint, or an error will be returned.*

-`quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-`verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<path_multiplier> - (Required) The number of clock cycles to move the setup and hold analysis analysis.

Note: When -`hold` is specified, the <path_multiplier> acts on the hold relationship to restore the hold check to its original position.

Examples

The following example establishes a path multiplier of 3 clock cycles for the setup check of the timing path defined by the -`from`/-`to` options. A path multiplier of N-1, or 2 in this example, is used to decrement the hold check on the same timing path:

```
set_multicycle_path 3 -setup -from [get_pins data0_reg/C] \
   -to [get_pins data1_reg/D]
set_multicycle_path 2 -hold -from [get_pins data0_reg/C] \
   -to [get_pins data1_reg/D]
```

Note: For more information on the relationship between the setup and hold analysis refer to the *Vivado Design Suite User Guide: Using Constraints (UG903).*

See Also

- report_timing
- report_timing_summary
- set_input_delay
- set_output_delay

set_operating_conditions

Set operating conditions for power estimation

Syntax

```
set_operating_conditions [-voltage <args>] [-grade <arg>] [-process <arg>]
    [-junction_temp <arg>] [-ambient_temp <arg>] [-thetaja <arg>]
    [-thetasa <arg>] [-airflow <arg>] [-heatsink <arg>] [-thetajb <arg>]
    [-board <arg>] [-board_temp <arg>] [-board_layers <arg>]
    [-design_power_budget <arg>] [-supply_current_budget <args>] [-quiet]
    [-verbose]
```

Usage

Name	Description
[-voltage]	List of voltage pairs, e.g., {name value}. Supported voltage supplies vary by family.
[-grade]	Temperature grade. Supported values vary by family. Default: commercial
[-process]	Process data: typical or maximum Default: typical
[-junction_temp]	Junction Temperature (C): auto|degC Default: auto
[-ambient_temp]	Ambient Temperature (C): default|degC Default: default
[-thetaja]	ThetaJA (C/W): auto|degC/W Default: auto
[-thetasa]	ThetaSA (C/W): auto|degC/W Default: auto
[-airflow]	Airflow (LFM): 0 to 750 Default: varies by family
[-heatsink]	Dimensions of heatsink: none, low, medium, high, custom Default: medium
[-thetajb]	ThetaJB (C/W): auto|degC/W Default: auto
[-board]	Board type: jedec, small, medium, large, custom Default: medium
[-board_temp]	Board Temperature degC
[-board_layers]	Board layers: 4to7, 8to11, 12to15, 16+ Default: 8to11
[-design_power_budget]	Design Power Budget (W) Default: Unspecified
[-supply_current_budget]	Sets list of supply current budget 'name value' pairs. Supported voltage supplies vary by family.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

SDC, XDC, Power

Description

Sets the real-world operating conditions that are used when performing analysis of the design. The environmental operating conditions of the device are used for power analysis when running the `report_power` command.

Note: This command operates silently and does not return direct feedback of its operation.

Operating conditions can be restored to their default values with the use of the `reset_operating_conditions` command.

Current operating conditions can be reported with the `report_operating_conditions` command.

Arguments

`-voltage <arg>` - (Optional) List of voltage supply names and their values specified in pairs. Supported voltage supply names and their values vary by family. For example if a family supports a voltage supply named Vccint, you can set the supply voltage to 0.8 with the following argument and value: `-voltage {Vccint 0.8}`

Note: If you specify a voltage that is outside the valid operating range for the target device, the `set_operating_conditions` command can change the device speedgrade to match the specified voltage. This can have an affect on timing analysis. For UltraScale devices, when changing the Vccint voltage, the Vivado tool will automatically change the device to or from a low-voltage device as indicated by the voltage level specified.

`-grade <arg>` - (Optional) The temperature grade of the target device. Supported values vary by family. The default value is "commercial".

`-process <arg>` - (Optional) The manufacturing process characteristics to assume. Valid values are "typical" or "maximum". The default value is "typical".

`-junction_temp <arg>` - (Optional) The device junction temperature used for modeling. Valid values are "auto" or an actual temperature specified in degrees C. The default value is "auto".

`-ambient_temp <arg>` - (Optional) The environment ambient temperature in degrees C. The default setting is "default".

`-thetaja <arg>` - (Optional) The Theta-JA thermal resistance used for modeling in degrees C/W. The default setting is "auto".

`-thetasa <arg>` - (Optional) The Theta-SA thermal resistance used during modeling in degrees C/W. The default setting is "auto".

`-airflow <[0:750]>` - (Optional) Linear Feet Per Minute (LFM) airflow to be used for modeling. The default setting varies by device family.

`-heatsink <arg>` - (Optional) The heatsink profile to be used during modeling. Valid values are: none, low, medium, high, custom. The default setting is "medium".

`-thetajb <arg>` - (Optional) The Theta-JB thermal resistance used for modeling in degrees C/W. The default setting is "auto".

`-board <arg>` - (Optional) The board size to be used for modeling. The valid values are: jedec, small, medium, large, custom. The default value is "medium".

`-board_temp <arg>` - (Optional) The board temperature in degrees Centigrade to be used for modeling.

`-board_layers <arg>` - (Optional) The number of board layers to be used for modeling. Valid values are: "4to7" for boards with 4 to 7 layers, "8to11" for boards with 8 to 11 layers, "12to15" for boards with 12 to 15 layers, and "16+" for boards with 16 or more layers. The default setting is "12to15".

`-design_power_budget <arg>` - (Optional) The design power budget in Watts. This value is used by the `report_power` command to report the difference between the calculated on-chip power and the design power budget. When unspecified, the difference is not reported. The default is "unspecified".

`-supply_current_budget <arg>` - (Optional) List of voltage supply names and their current budget values specified in pairs. Supported voltage supply names vary by family. For example if a family supports a voltage supply named Vccint, you can set the supply current budget to 10 A with the following argument and value: `-supply_current_budget {Vccint 10}`

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example specifies an industrial grade device with an ambient operating temperature of 75 degrees C:

```
set_operating_conditions -grade industrial -ambient_temp 75
```

The following example sets the supply voltage Vccaux to a value of 1.9:

```
set_operating_conditions  -voltage {Vccaux 1.89}
```

Send Feedback

The following example sets the manufacturing process corner to maximum:

```
set_operating_conditions -process maximum
```

The following example sets the manufacturing process corner to maximum and the voltage supply Vccint to 0.875:

```
set_operating_conditions -process maximum -voltage {Vccint 0.875}
```

See Also

- report_operating_conditions
- report_power
- reset_operating_conditions

set_output_delay

Set output delay on ports

Syntax

```
set_output_delay [-clock <args>] [-reference_pin <args>] [-clock_fall]
    [-rise] [-fall] [-max] [-min] [-add_delay] [-network_latency_included]
    [-source_latency_included] [-quiet] [-verbose] <delay> <objects>
```

Usage

Name	Description
[-clock]	Relative clock
[-reference_pin]	Relative pin or port
[-clock_fall]	Delay is relative to falling edge of clock
[-rise]	Specifies rising delay
[-fall]	Specifies falling delay
[-max]	Specifies maximum delay
[-min]	Specifies minimum delay
[-add_delay]	Don't remove existing input delay
[-network_latency_included]	Specifies network latency of clock already included
[-source_latency_included]	Specifies source latency of clock already included
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<delay>	Delay value
<objects>	List of ports

Categories

[SDC](), [XDC]()

Description

> 💡 **TIP:** *The XDC > Timing Constraints language templates and the Timing Constraints Wizard in the Vivado IDE offer timing diagrams and additional details around defining specific timing constraints. You can refer to these sources for additional information.*

Specifies the external system-level path delay on a primary output port relative to a clock edge at the interface of the design. The output delay value is specified in nanoseconds (ns), and can be positive or negative, depending on the clock and data relative phase outside the FPGA device.

To accurately model the system-level timing of your Xilinx FPGA design, you must assign timing delays for objects external to the FPGA onto the primary input or output ports in your design. These delays are defined by the `set_input_delay` and `set_output_delay` commands.

> **IMPORTANT!** *If the output port also has a `set_max_delay` constraint assigned, the specified output delay value is considered part of the max_delay computation. That is, the output delay consumes a portion of the max delay on the timing path that includes the output port.*

This command returns nothing if successful, or returns an error if it fails.

Arguments

`-clock <arg>` - (Optional) Indicates that the delay is relative to the rising edge of the specified clock.

`-reference_pin <arg>` - (Optional) Specifies that the delay is relative to the specified pin rather than a clock.

`-clock_fall` - (Optional) Specifies that the delay is relative to a falling edge of the clock rather than rising edge.

`-rise` - (Optional) Specifies that the delay is for a rising edge.

`-fall` - (Optional) Specifies that the delay is for a falling edge

`-max` - (Optional) Specifies that the delay specified should be treated as a maximum threshold.

`-min` - (Optional) Specifies that the delay specified should be treated as a minimum threshold.

`-add_delay` - (Optional) Add the specified delay constraint to the port, to coexist with any other `set_output_delay` constraints already defined on the port. The default behavior is to replace the existing delays.

`-network_latency_included` - (Optional) Indicates that the clock network latency of the reference clock is included in the delay value. The Vivado timing engine considers the clock edge reaching the capture flop after the clock latencies unless the specified input or output delay value includes the source latency or network latency.

`-source_latency_included` - (Optional) Specifies that the source latency of the reference clock is included in the specified delay value.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<delay> - (Optional) The delay specified in nanoseconds (ns) to apply to the listed ports. Valid values are floating point numbers, with a default value of 0.

`<objects>` - (Required) A list of ports to which the delay applies.

Examples

The following example sets an output delay on ports relative to the specified clock:

```
set_output_delay 5.0 -clock [get_clocks cpuClk] [get_ports]
```

The next example is the same as the prior example except that network latency is now included:

```
set_output_delay 5.0 -clock [get_clocks cpuClk] \
    -network_latency_included [get_ports]
```

This example creates a clock named clk_ddr, and defines output delay constraints from data launched by both rising and falling edges of the clock outside the device to the data output of the internal flip-flop that is sensitive to both rising and falling clock edges:

```
create_clock -name clk_ddr -period 6 [get_ports DDR_CLK_IN]
set_output_delay -clock clk_ddr -max 2.1 [get_ports DDR_OUT]
set_output_delay -clock clk_ddr -max 1.9 [get_ports DDR_OUT] -clock_fall -
add_delay
set_output_delay -clock clk_ddr -min 0.9 [get_ports DDR_OUT]
set_output_delay -clock clk_ddr -min 1.1 [get_ports DDR_OUT] -clock_fall -
add_delay
```

Note: The use of the `-add_delay` option allows the new min and max delay constraints to exist alongside the first delays on the same port.

See Also

- all_clocks
- create_clock
- get_ports
- set_max_delay
- set_input_delay

set_package_pin_val

Set user columns on one or more package pins

Syntax

```
set_package_pin_val [-quiet] [-verbose] <column> <value> <package_pins>...
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<column>	User column name
<value>	Value to set
<package_pins>	Package pin names

Categories

XDC, PinPlanning

Description

Create user-defined package pin attributes and assign values to specific pins on the package.

User-defined pin attributes can be defined in a CSV file and imported into an I/O Pin Planning project using `read_csv`, or can be edited in the project using this command.

Note: Use the `set_property` command to set tool-defined properties of a package pin.

Arguments

<column> - (Required) Specify the user-defined column name. The column name is case-sensitive. If the column does not already exist, a new attribute is created for package pins. If the user-defined column name already exists, the specified value is assigned to the specified pins.

Note: Column refers to the display of the attribute in the Package Pins view in the tool GUI. The result of the command is an attribute on the specified package pins that can be exported with `write_csv` for instance.

`<value>` - (Required) Specify the value to assign to the specified column. You can repeat the `set_package_pin_val` command to assign different values to different pins in the same column.

<package_pins> - (Required) Specify the package pins to assign the value to. You can use the `get_package_pins` command to specify the package pins to set.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example creates a new user-defined column in the Package Pins view, and assigns the value `true` to the specified pin:

```
set_package_pin_val -column track1 -value true -package_pins AK27
```

The following example creates a user-defined column called `Test`, then assigns the value RED to all "AK" package pins, then changes the value to GREEN for the three specified pins:

```
set_package_pin_val -column Test -value RED \
    -package_pins [get_package_pins AK*]
set_package_pin_val -column Test -value GREEN \
    -package_pins {AK1 AK2 AK3}
```

See Also

- get_package_pins
- set_property
- write_csv

set_param

Set a parameter value

Syntax

```
set_param [-quiet] [-verbose] <name> <value>
```

Returns

Newly set parameter value.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<name>	Parameter name
<value>	Parameter value

Categories

PropertyAndParameter

Description

Sets the value of a user-definable configuration parameter. These parameters configure and control various behaviors of the tool. Refer to `report_param` for a description of currently defined parameters.

As an example, a specific param that can be defined is the `general.maxThreads` parameter for the Vivado Design Suite. On multiprocessor systems, the Vivado Design Suite use multi-threading to speed up certain processes, including DRC reporting, static timing analysis, placement, and routing. A default limit applies to all tasks and is based on the operating system. For Windows systems, the default is 2; for Linux systems the default is 8. The limit can be changed as follows:

```
set_param general.maxThreads <value>
```

Where `<value>` is an integer from 1 to 8, inclusive.

The maximum number of simultaneous threads that can be used also varies by the task being run. You can change the `maxThreads` parameter prior to running these processes. The maximum number of threads for specific Tcl commands are:

- phys_opt_design: 8

- place_design: 8

- report_drc: 8

- report_timing and report_timing_summary: 8

- route_design: 8

- synth_design: 4

You can use the `reset_param` command to restore any parameter that has been modified back to its default setting.

Note: Setting a specified parameter value to -1 will disable the feature.

Arguments

`<name>` - (Required) The name of the parameter to set the value of. You can only set the value of one parameter at a time.

`<value>` - (Required) The value to set the specified parameter to.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example sets the parameter defining how many threads to run for multi-threaded processes, including Placement, Routing, and Timing Analysis:

```
set_param general.maxThreads 4
```

Note: The Vivado tool supports between 1 to 8 threads. Use `get_param` to determine the current setting.

The following example sets a new default value for message limit:

```
set_param messaging.defaultLimit 1000
```

See Also

- get_param

- [list_param](#)
- [report_param](#)
- [reset_param](#)

Send Feedback

set_part

Sets the part on the current project. If no project is open, then a diskless project is created.

Syntax

```
set_part [-quiet] [-verbose] <part>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<part>	Set current project's part to this part.

Categories

Project, PropertyAndParameter

Description

Change the part used by the current project for subsequent elaboration, synthesis, implementation, and analysis.

> **TIP:** *The part is changed for the current project only, and not for the in-memory design. You can change the speed grade of the device in the in-memory design for timing analysis using the `set_speed_grade` command. You can change the part used when opening an existing design checkpoint using the `-part` option of the `open_checkpoint` or `read_checkpoint` commands.*

This command is provided to let you change the part for the in-memory project of non-project based designs, and does not support project-based designs. For a project-based design set the PART property on the project as follows:

```
set_property PART xc7vx485tffg1158-2 [current_project]
```

Use the `get_parts` command to get a list of the available parts.

The `set_part` command creates an in-memory project for a non-project based design, or assigns the part to the existing in-memory project.

Note: For a discussion of Project Mode and Non-Project Mode refer to the *Vivado Design Suite User Guide: Design Flows Overview* (UG892).

This command returns the part that the in-memory project is set to use, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<part> - (Required) Specifies the part to change to, or use in the current project or in-memory design.

Example

The following example changes the part of the current in-memory project:

```
set_part xc7vx485tffg1158-2
```

See Also

- create_project
- get_parts
- open_checkpoint
- read_checkpoint
- set_property
- set_speed_grade

Send Feedback

set_power_opt

Set constraints for power optimization

Syntax

```
set_power_opt [-include_cells <args>] [-exclude_cells <args>]
    [-clocks <args>] [-cell_types <args>] [-quiet] [-verbose]
```

Usage

Name	Description
[-include_cells]	Include only these instances for clock gating. Default: all
[-exclude_cells]	Exclude these instances from clock gating. Default: none
[-clocks]	Clock gate instances clocked by these clocks only. Default: all clocks
[-cell_types]	Clock gate these cell types only. Specify either [all|none], or one or more of [bram|reg|srl]. Default: all
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Power](#), [XDC](#)

Description

Specify cell instances to include or exclude in power optimization. The specified cells are optimized using the `power_opt_design` command.

> 💡 **TIP:** *Block RAM optimizations are performed by default with the* `opt_design` *command. Some or all BRAM cells can be excluded from the* `opt_design` *optimization using the* `set_power_opt` *command as well.*

The effect of multiple `set_power_opt` commands is cumulative, so that you can specify a broad class of cell types to optimize, include specific hierarchical cells, and then exclude cells within the included hierarchy to refine the power optimization.

The power optimizations that have been performed can be reported using the `report_power_opt` command.

Arguments

`-include_cells` <args> - (Optional) Include only these instances for clock gating. Use this option to list specific cells or blocks to be optimized using `power_opt_design`. The default is to include all cells in power optimization.

`-exclude_cells` <args> - (Optional) Exclude these instances from clock gating. The default is to not exclude cells from power optimization. The `-exclude_cells` option excludes from the currently included cells. By default all cells are included, however, if `-include_cells` has been specified, then `-exclude_cells` applies only to the currently included cells.

`-clocks` <args> - (Optional) Perform power optimizations on instances clocked by the specified clocks only. The default is to include all clocks in the design.

Note: It is possible to use both `-clocks` and `-include_cells` to produce a list of cells that are not clocked by the specified clocks, resulting in no power optimization.

`-cell_types [ all | bram | reg | srl | none ]` - (Optional) Perform power optimization on the specified cell types only. The default is to perform power optimization on all types of cells. You can use `all` or `none` to reset, or clear, any prior `set_power_opt` commands. You can also specify one or more of bram, srl, or reg type cells.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example sets power optimization for BRAM cells only, and then runs power optimization:

```
set_power_opt -cell_types bram
power_opt_design
```

The following example sets power optimization for BRAM and REG type cells, then adds SRLs, and runs power optimization. Then all cells are cleared, and only SRLs are included, and power optimization is run again:

```
set_power_opt -cell_types { bram reg}
set_power_opt -cell_types { srl}
power_opt_design
set_power_opt -cell_types { none}
set_power_opt -cell_types { srl}
power_opt_design
```

The following example sets power optimization for BRAM cells only, excludes the cpuEngine block from optimization, but then includes the cpuEngine/cpu_dbg_dat_i block, then performs power optimization:

```
set_power_opt -cell_types bram
set_power_opt -exclude_cells cpuEngine
set_power_opt -include_cells cpuEngine/cpu_dbg_dat_i
power_opt_design
```

See Also

- opt_design
- power_opt_design
- report_power_opt

set_propagated_clock

Specify propagated clock latency

Syntax

```
set_propagated_clock [-quiet] [-verbose] <objects>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<objects>	List of clocks, ports, or pins

Categories

SDC, XDC

Description

Propagates clock latency throughout a clock network, resulting in more accurate skew and timing results throughout the clock network.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<objects>` - (Required) A list of the clock objects to force propagation on.

Examples

This example specifies that the primary system clock from the top-level should be propagated:

```
set_propagated_clock [get_clocks top/clk]
```

This example specifies that all clocks from "sublevel1" should be propagated:

```
set_propagated_clock [get_clocks sublevel1/*]
```

See Also

- get_clocks
- create_clock

Send Feedback

set_property

Set property on object(s)

Syntax

```
set_property [-dict <args>] [-quiet] [-verbose] <name> <value> <objects>...
```

Usage

Name	Description
[-dict]	list of name/value pairs of properties to set
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<name>	Name of property to set. Not valid with -dict option
<value>	Value of property to set. Not valid with -dict option
<objects>	Objects to set properties on

Categories

Object, PropertyAndParameter, XDC

Description

Assigns the defined property `<name>` and `<value>` to the specified `<objects>`.

This command can be used to define any property on an object in the design. Each object has a set of predefined properties that have expected values, or a range of values. The set_property command can be used to define the values for these properties. To determine the defined set of properties on an object, use `report_property`, `list_property`, or `list_property_values`.

You can also define custom properties for an object, by specifying a unique `<name>` and `<value>` pair for the object. If an object has custom properties, these will also be reported by the `report_property` and `list_property` commands.

This command returns nothing if successful, and an error if it fails.

> 💡 **TIP:** *You can use the `get_property` command to validate any properties that have been set on an object.*

Send Feedback

Arguments

`-dict` - (Optional) Use this option to specify a dictionary of multiple properties (`<name>` `<value>` pairs) on an object with a single `set_property` command. Multiple `<name>` `<value>` pairs must be enclosed in quotes, "", or braces, {}.

```
-dict "name1 value1 name2 value2 ... nameN valueN"
```

> **IMPORTANT!** *When writing the constraints for a design using either `save_constraints`, `save_constraints_as`, or `write_xdc`, the properties specified using the `-dict` option will be written as separate `set_property` commands for each name/value pair. If you don't want the XDC constraints to be expanded in this manner, you can either use the Tcl script driven approach in a non-project design, or use a Tcl script as a design source in your constraint set. Refer to Vivado Design Suite User Guide: Design Flows Overview (UG892) for more information on non-project based design, or refer to Vivado Design Suite User Guide: Using Constraints (UG903) for more information on using Tcl scripts in constraint sets.*

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<name>` - (Required) Specifies the name of the property to be assigned to the object or objects. The `<name>` argument is case sensitive and should be specified appropriately.

`<value>` - (Required) Specifies the value to assign to the `<name>` on the specified object or objects. The value is checked against the property type to ensure that the value is valid. If the value is not appropriate for the property an error will be returned.

> **IMPORTANT!** *In some cases the value of a property may include special characters, such as the dash character ('-'), which can cause the tool to interpret the value as a new argument to the command. In this case, you must use the explicit arguments (`-name`, `-value`, `-objects`) instead of the implied positional arguments (name, value, objects) as described here. This is shown in the Examples section below.*

`<objects>` - (Required) One or more objects to assign the property to.

Examples

Create a user-defined boolean property, TRUTH, for cell objects, and set the property on a cell:

```
create_property -type bool truth cell
set_property truth false [lindex [get_cells] 1]
```

Use the -dict option to specify multiple properties at one time on the current design:

```
set_property -dict "POST_CRC enable POST_CRC_ACTION correct_and_continue" \
[current_design]
```

The following example sets the TOP property of the current fileset to define the top module of the project:

```
set_property top fftTop [current_fileset]
set_property top_file {C:/Data/sources/fftTop.v} [current_fileset]
```

Note: Defining the top module requires the TOP property to be set to the desired hierarchical block in the source fileset of the current project. In the preceding example TOP is the property name, fftTop is the value, and current_fileset is the object. In addition, the TOP_FILE property should be defined to point to the data source for the top module.

This example shows how to set a property value that includes the dash character, '-'. The dash can cause the tool to interpret the value as a new command argument, rather than part of the value being specified, and will cause an error as shown. In this case, you must use the explicit form of the positional arguments in the set_property command:

```
set_property {XELAB.MORE_OPTIONS} {-pulse_e_style ondetect} \
   [get_filesets sim_1]
ERROR: [Common 17-170] Unknown option '-pulse_e_style ondetect',
 please type 'set_property -help' for usage info.
set_property -name {XELAB.MORE_OPTIONS} -value {-pulse_e_style ondetect}\
   -objects [get_filesets sim_1]
```

The following example sets the internal VREF property value for the specified IO Bank:

```
set_property internal_vref {0.75} [get_iobanks 0]
```

The following example defines a DCI Cascade by setting the DCI_CASCADE property for the specified IO Bank:

```
set_property DCI_CASCADE {14} [get_iobanks 0 ]
```

The following example configures the synth_1 run, setting options for Vivado Synthesis 2013, and then launches the synthesis run:

```
set_property flow {Vivado Synthesis 2016} \
   [get_runs synth_1]
set_property STEPS.SYNTH_DESIGN.ARGS.GATED_CLOCK_CONVERSION on \
   [get_runs synth_1]
set_property STEPS.SYNTH_DESIGN.ARGS.FSM_EXTRACTION one_hot \
   [get_runs synth_1]
launch_runs synth_1
```

This example is the same as the prior example, except that it uses the -dict option to set all the properties on the synthesis run in a single set_property command:

```
set_property -dict [ list flow {Vivado Synthesis 2016} \
   STEPS.SYNTH_DESIGN.ARGS.GATED_CLOCK_CONVERSION on \
   STEPS.SYNTH_DESIGN.ARGS.FSM_EXTRACTION \
   one_hot ] [get_runs synth_1]
launch_runs synth_1
```

See Also

- current_fileset
- create_property
- create_run
- get_cells
- get_property
- get_runs
- launch_runs
- list_property
- list_property_value
- report_property
- reset_property

Send Feedback

set_speed_grade

Set Timing Speed Grade and Temperature Grade

Syntax

```
set_speed_grade [-temperature <arg>] [-quiet] [-verbose] [<value>]
```

Returns

String result

Usage

Name	Description
[-temperature]	Temperature grade used for timing analysis (Not available for 7 Series and earlier)
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<value>]	Speed grade used for timing analysis

Categories

[Project](#)

Description

Note: After `set_speed_grade` has been used on a design, it can be used for timing analysis, but it will no longer go through implementation. If you want to run implementation on the design, you should save the design checkpoint and use `read_checkpoint -part` to implement the design with the new speed grade.

Sets the speed grade used for timing analysis for the target device in the current design.

This command is used to change the speed grade of the target device for timing analysis only, and does not affect other aspects of the design. It must be run on an opened synthesized or implemented design.

Use the `set_speed_grade` command prior to the `report_timing_summary` or `report_timing` command or other timing commands to change the speed grade for analysis. If the timing is valid, then you can use the `set_property` or `set_part` command to change the target part for the project to re-synthesize and implement the design.

> **TIP:** *For UltraScale devices, you can specify either the temperature or the value to define the speed grade for the part. For 7 series devices, you can only specify the value.*

This command returns a transcript of its process, and the speed grade set, or returns an error if it fails.

Arguments

-temperature - (Optional) Specify the temperature grade for UltraScale devices, used for timing analysis.

Note: This option is not available for 7 series devices, or earlier.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<value>` - (Optional) The speed grade for the target device. Typical values are -1, -1L, -2, or -3. Specifying an incorrect value (e.g. -999) will cause the command to return an error message with a list of valid values for the current part.

Examples

The following example sets the speed grade for the device in the current design to -1:

```
set_speed_grade -1
```

See Also

- report_drc
- report_timing
- report_timing_summary
- set_part
- set_property

Send Feedback

set_switching_activity

Set switching activity on specified objects or default types

Syntax

```
set_switching_activity [-toggle_rate <arg>] [-default_toggle_rate <arg>]
    [-type <args>] [-all] [-static_probability <arg>]
    [-default_static_probability <arg>] [-signal_rate <arg>] [-hier]
    [-deassert_resets] [-quiet] [-verbose] [<objects>...]
```

Usage

Name	Description
[-toggle_rate]	Toggle rate (%) is the rate at which the output of synchronous logic element switches compared to a given clock input. It is modeled as a percentage between 0 - 200%. A toggle rate of 100% means that on average the output toggles once during every clock cycle, changing on either the rising or falling clock edges, and making the effective output signal frequency half of the clock frequency. Default: 0.0
[-default_toggle_rate]	The default toggle rate to be used in power analysis on the primary inputs of the design. The default toggle rate is set on those primary input nets whose switching activity is not specified by the user, simulation data or constraints of the design. Valid values are: 0 <= value < 200. The default value is 12.5. Default: 12.5
[-type]	Specify nodes in a specific category. List of valid type values: io_output, io_bidir_enable, register, lut_ram, lut, dsp, bram_enable, bram_wr_enable, gt_txdata, gt_rxdata.
[-all]	Used together with -type, set switching activity on -type nets within an instance
[-static_probability]	Static probability value: 0 <= Value <= 1 Default: 0.5
[-default_static_probability]	The default static probability to be used in power analysis on the design. The default static probability is set on those primary inputs whose switching activity is not specified by the user, simulation data or constraints of the design. Valid values are: 0 <= Value <= 1. The default value is 0.5. Default: 0.5
[-signal_rate]	The number of times an element changed state (high-to-low and low-to-high) per second. Xilinx tools express this as millions of transitions per second (Mtr/s). Default: 0.0
[-hier]	Hierarchically sets the switching activity on a hierarchical instance provided via <objects> option. This option should be used only with <objects> option
[-deassert_resets]	A switch to elegantly auto deassert all set,reset,preset and clear signals that do not have conflicted polarities
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<objects>]	Objects to set switching activity on

Send Feedback

Categories

XDC, Power

Description

Sets the signal rate and the switching probability to be used when performing power estimation on the current synthesized or implemented design. These include simple signal rate and simple static probability on nets, ports, and pins; and state dependent static probabilities on cells.

Note: This command operates silently and does not return direct feedback of its operation.

The switching activity of a design affects both the static and dynamic power consumption. The static power is often dependent on logic state transitions, and the dynamic power is directly proportional to the toggle rate.

The `set_switching_activity` command can be used to specify default activity rates for the whole design, or to define the activity of one or more signals in the design or on a specified module.

The current switching activity attributes can be found by using the `report_switching_activity` command. The values can be set to their default values by using the `reset_switching_activity` command.

Note: The `reset_switching_activity` is used to reset switching activity for specified objects. Use the `set_switching_activity -default_toggle_rate` or `-default_static_probability` to change or reset these values.

Arguments

`-toggle_rate <value>` - (Optional) Specify the toggle rate (%) as the switching rate of the output of synchronous logic elements compared to a given clock input. The toggle rate is specified as a percentage between 0 - 100%. For clock and DDR signals only, the toggle rate can be specified up to 200%. A toggle rate of 100% means that on average the output toggles once during every clock cycle, changing on either the rising or falling clock edges, and making the effective output signal frequency half of the clock frequency. The default `<value>` is 12.5% of the clock frequency, as defined by the `-default_toggle_rate` option.

Note: This option must be specified with `-static_probability`.

`-default_toggle_rate <rate>` - (Optional) The default toggle rate to be used in power analysis on the primary inputs of the current design. The default toggle rate is set on the primary input nets whose switching activity is not specified by the user, simulation data, or constraints of the design. On asynchronous inputs the toggle rate is set with respect to the highest clock in the design. Valid values are: 0 <= value < 200. The default value is 12.5.

Send Feedback

Note: This option cannot be specified with `<objects>` as it specifies the default toggle rate for the entire design.

`-type <arg>` - (Optional) The type of logic entity that the specified `set_switching_activity` command can be applied to. By default, the command is applied to the top-level of the current design, or to the specified `<objects>`. The `-type` option applies the command settings to the specified type of logic objects in the top-level of the current design. The `-all` option or `-hier` option can be used to modify the scope of objects the command applies to. Valid logic types include:

- io_output - Primary outputs.

- io_bidir_enable - Enable pin of Bidir ports.

- register - All register outputs in the design/hierarchy specified.

- lut - All LUT outputs in the design/hierarchy specified.

- lut_ram - All distributed ram outputs in the design/hierarchy specified.

- dsp - All DSP outputs in the design/hierarchy specified.

- bram_enable - Enable pins (ENARDEN/ENBWREN) of BRAMs.

- bram_wr_enable - Write enables of BRAMs (WEA/WEBWE).

- gt_txdata - Output TX data pins of all GTs.

- gt_rxdata - Output RX data pins of all GTs.

Note: The `-type` option is only valid for use when setting the `-toggle_rate` or the `-signal_rate`, and other settings are not supported.

`-all` - (Optional) Must be used with `-type` to set the switching activity on all instances in the design of the specified type of logic objects. As a default the `set_switching_activity` command applies to the top-level of the design or current_instance, or to the specified type of objects in that level.

`-static_probability <value>` - (Optional) The static switching probability for the specified `<objects>` to be used in power analysis. Valid values are 0 <= `<value>` <= 1. The default value is 0.5.

`-default_static_probability <value>` - (Optional) The default static probability to be used in power analysis on the current design. The default static probability is set on primary inputs whose switching activity is not specified by the user, simulation data, or constraints of the design. Valid values are 0 <= `value` <= 1. The default value is 0.5.

Note: This option cannot be specified with `<objects>` as it defines the default for the entire design. Use `-static_probability` to define the `<value>` for specific objects.

`-signal_rate <value>` - (Optional) The signal frequency to be used for analysis, expressed as millions of transitions per second (Mtr/s). Specifies the number of times an element changes state per second, including transitions from low-to-high and from high-to-low. The default value is 0.

Note: Must be specified with `-static_probability`.

`-hier` - (Optional) Apply the switching activity hierarchically to signals in the specified hierarchical `<objects>`. Without `-hier`, the switching activity is applied to the specified `<objects>`, or `-type` of objects at the current level of the hierarchy.

`-deassert_resets` - (Optional) Directs the Vivado tool to automatically de-assert all reset-type control signals (Set, Preset, Reset, Clear), by setting the signal static probability to 0. The tool will only de-assert control signals with non-conflicting polarities. However, if a net is connected to an active-Low and an active-High Reset pin for instance, then it will not de-assert this signal; or if a net is connected to an active-High Reset, and an active-High Set, or Enable pin, then it will not de-assert this signal.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<objects>` - (Optional) A list of port, pin, and net objects to which the switching activity constraint should be applied; or a list of cells when specified with `-type` to define logic objects.

Examples

The following example specifies a signal rate and switching probability for all ports, then reports the switching attributes for those ports:

```
set_switching_activity -signal_rate 55 -static_probability .33 [get_ports]
report_switching_activity [get_ports]
```

The following example specifies the default switching probability for the current design:

```
set_switching_activity -default_static_probability .75
```

This example sets the specified toggle rate and static probability on all registers in the hierarchy of "CPU/MEM":

```
set_switching_activity -type register -toggle_rate 0.4 \
    -static_probability 0.5 [get_cells CPU/MEM]
```

This example sets the specified toggle rate and static probability on all registers in the hierarchy of "CPU/" and underneath hierarchy:

```
set_switching_activity -type register -toggle_rate 0.4
    -static_probability 0.5 -hier [get_cells CPU]
```

See Also

- get_clocks
- get_nets
- get_ports
- power_opt_design
- report_power
- report_switching_activity
- reset_switching_activity

Send Feedback

set_system_jitter

Set system jitter

Syntax

```
set_system_jitter [-quiet] [-verbose] <system_jitter>
```

Returns

System_jitter

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<system_jitter>	System jitter: Value >= 0

Categories

[XDC](#)

Description

Sets the system jitter specified in nanoseconds (ns) for all clocks in the design, including primary and generated clocks. System jitter is used to account for excessive noise that affects all the clocks within the FPGA, like power supply noise and board noise. The default system jitter is technology-dependent and is predefined for each Xilinx FPGA family based on device characterization with several power supplies under all supported operating conditions.

System Jitter is a component of the Total System Jitter (T_{sj}) used in the calculation of clock uncertainty for a path. It is due to the maximum noise (in time) that can be seen on the V_{ccint} rail due to simultaneous switching of internal nodes, cross talk and other phenomenon that can impact timing on any path in the design.

> ⭐ **IMPORTANT!** *The jitter calculated by Xilinx takes into consideration the uncertainty introduced by the clocking resources, the input jitter and the system jitter. Using the* `set_system_jitter` *command overrides the default system jitter value calculated by Xilinx, and is not recommended.*

The System Jitter and the Input Jitter are random jitters which typically follow a Gaussian distribution and are added in a quadratic manner to represent the worst case combination. When the Input Jitter is null, the Total System Jitter (T_{sj}) for an internal register-to-register path has the following equation:

- $T_{sj} = \sqrt{(SourceClockSystemJitter^2 + DestinationClockSystemJitter^2)}$

For example, when using the default value for system jitter of 50ps:

- $T_{sj} = \sqrt{(0.050^2 + 0.050^2)} = 0.071ns = 71ps$

The `set_system_jitter` command applies to all the clocks in the design. Use the `set_input_jitter` command to specify additional jitter for a specific primary clock.

> 💡 **TIP:** *SYSTEM_JITTER is reported as a property of clocks, although it applies to all clocks in the design. INPUT_JITTER is also a property of primary clocks. These properties can be returned by the* `get_property` *or* `report_property` *commands.*

This command returns nothing if successful, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<system_jitter> - (Required) Specifies the system jitter specified in nanoseconds (ns) to be applied system-wide. The jitter specified by the `set_system_jitter` command overwrites the default value.

Examples

This example defines the primary clock, sysClk, and specifies a system wide jitter of 0.1 ns:

```
create_clock -period 10 -name sysClk [get_ports sysClk]
set_system_jitter 0.1
```

The following example defines a primary clock, sysClk, and a generated clock, sysClkDiv2, that is a divide by two version of the primary clock. A system jitter of 0.2 ns is specified that applies to all the clocks in the design. An additional input jitter of 0.09 ns is specified on only the primary clock:

```
create_clock -period 10 -name sysClk [get_ports sysClk]
create_generated_clock  -name sysClkDiv2 -source [get_ports sysClk] \
   -divide_by 2 [get_pins clkgen/sysClkDiv/Q]
set_system_jitter 0.2
set_input_jitter sysClk 0.09
```

Send Feedback

The follow example defines two primary clocks, sysClk and procClk. A system jitter of 0.2 ns is defined for all the clocks in the system. An additional input jitter of 0.05 ns is specified for the clock procClk:

```
create_clock -period 10 -name sysClk [get_ports sysClk]
create_clock -period 25 -name procClk [get_ports procClk]
set_system_jitter 0.2
set_input_jitter procClk 0.05
```

See Also

- report_timing
- set_clock_uncertainty
- set_input_delay
- set_input_jitter

set_units

Set units for checking

Syntax

```
set_units [-capacitance <arg>] [-current <arg>] [-voltage <arg>]
    [-power <arg>] [-resistance <arg>] [-altitude <arg>] [-quiet]
    [-verbose]
```

Usage

Name	Description
[-capacitance]	Capacitance unit in farad. Valid values are from kF-fF. Default: pF
[-current]	Current unit in ampere. Valid values are from kA-fA. Default: mA
[-voltage]	Voltage unit in volt. Valid values are from kV-fV. Default: V
[-power]	Wattage unit in watts. Valid values are from kW-fW. Default: mW
[-resistance]	Resistance unit in ohm. Valid values are from kOhm-fOhm. Default: ohm
[-altitude]	Altitude in metric or standard units. Valid values are meters and feet. Default: meters
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

SDC, XDC

Description

This command specifies the default units to be assumed when the design is analyzed. Specifically, the `-current`, `-voltage`, `-power`, and `-resistance` options impact the values returned by the `report_power` command.

The `set_units` command can be used multiple times to define and redefine units. If `set_units` includes a previously set unit value, the unit is redefined.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

`-capacitance <arg>` - (Optional) Specify the unit of capacitance in Farads. Valid values range from kilofarads (kF) to femtofarads (fF). The default unit of capacitance is picofarads (pF).

`-current <arg>` - (Optional) Specify the unit of current in amperes. Valid values range from kiloAmps (kA) to femtoAmps (fA). The default unit of amperes is milliAmps (mA).

`-voltage <arg>` - (Optional) Specify the unit of voltage in Volts. Valid values range from kilovolts (kV) to femotovolts (fV). The default unit of voltage is Volts (V).

`-power <arg>` - (Optional) Specify the unit of power in watts. Valid values range from kilowatts (kW) to femtowatts (fW). The default unit of power is milliwatts (mW).

`-resistance <arg>` - (Optional) Specify the unit of resistance in ohms. Valid values range from kilo-ohm (kOhm) to femto-ohm (fOhm). The default unit of resistance is ohms (Ohm).

`-altitude [ meters | feet ]` - (Optional) Specify the unit of altitude as `meters` or `feet`. The default unit is `meters`.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

Specify that voltage should be in millivolts and all values should use three digits

```
set_units -voltage mV
```

The following example changes the default unit for current to Amperes:

```
set_units -voltage kV -current A
```

Note: The second example of `set_units` redefines the Voltage units defined in the first example, as well as defining the units for current.

See Also

- report_power
- set_operating_conditions

set_value

Set the current value of an HDL object (variable, signal, wire, or reg) to a specified value

Syntax

```
set_value [-radix <arg>] [-quiet] [-verbose] <hdl_object> <value>
```

Usage

Name	Description
`[-radix]`	radix specifies the radix to use for interpreting value. Allowed values are: default, dec, bin, oct, hex, unsigned, ascii, smag
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<hdl_object>`	Set the value on the given hdl_object.
`<value>`	The value to assign to the specified object.

Categories

Simulation

Description

Specify the value of a single HDL object at the current simulation run time.

HDL objects include HDL signals, variables, or constants as defined in the Verilog or VHDL test bench and source files. An HDL signal includes Verilog wire or reg entities, and VHDL signals. Examples of HDL variables include Verilog real, realtime, time, and event.

HDL constants include Verilog parameters and localparams, and VHDL generic and constants. The HDL scope, or scope, is defined by a declarative region in the HDL code such as a module, function, task, process, or begin-end blocks in Verilog. VHDL scopes include entity/architecture definitions, block, function, procedure, and process blocks.

Arguments

`-radix <arg>` - (Optional) Specifies the radix to use when returning the value of the specified object. Allowed values are: `default`, `dec`, `bin`, `oct`, `hex`, `unsigned`, `ascii`, or `smag`.

Note: The radix `dec` indicates a signed decimal. Specify the radix `unsigned` when dealing with unsigned data.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hdl_object> - (Required) Specifies a single HDL object to get the value of. The object can be specified by name, or can be returned as an object from the `get_objects` command.

`<value>` - (Required) The value to set the specified object to. The specified `<value>` depends on the type of the <hdl_object>. HDL object types include: "logic", floating point, VHDL enumerated, and VHDL integral. For all but "logic" the `-radix` option is ignored.

- "Logic" does not refer to an actual HDL object type, but means any object whose values are similar to those of VHDL std_logic, such as:

  - the Verilog implicit 4-state bit type,

  - the VHDL bit and std_logic predefined types,

  - any VHDL enumeration type which is a subset of std_logic, including the character literals '0' and '1'.

- For logic types the value depends on the radix:

  - If the specified value has fewer bits than the logic type expects, the value is zero extended, but not sign extended, to match the expected length.

  - If the specified value has more bits than the logic type expects, the extra bits on the MSB side should all be zeros, or the Vivado simulator will return a "size mismatch" error.

- Accepted values for floating point objects are floating point values.

- The accepted value for non-logic VHDL enumerated types is a scalar value from the enumerated set of values, without single quotes in the case of characters.

- Accepted values for VHDL integral types is a signed decimal integer in the range accepted by the type.

Examples

The following example sets the value of the sysClk signal:

```
set_value sysClk Z
```

This example uses the `bin`, `dec`, and `unsigned` radix to specify the same value on the given bus:

```
set_value -radix bin /test/bench_VStatus_pad_0_i[7:0] 10100101
set_value -radix unsigned /test/bench_VStatus_pad_0_i[7:0] 165
set_value -radix dec /test/bench_VStatus_pad_0_i[7:0] -91
```

The following example shows the bit extension performed when the provided value has fewer bits than the logic type expects:

```
set_value -radix bin /test/bench_VStatus_pad_0_i[7:0] 101
get_value -radix bin /test/bench_VStatus_pad_0_i[7:0]
   00000101
```

The following example shows the bit truncation performed when the provided value has more bits than the logic type expects:

```
set_value -radix bin /test/bench_VStatus_pad_0_i[7:0] 0010100101
get_value -radix bin /test/bench_VStatus_pad_0_i[7:0]
   10100101
set_value -radix bin /test/bench_VStatus_pad_0_i[7:0] 1110100101
ERROR: [#UNDEF] Object size 8 does not match size of given value 1110100101
```

Note: In the second `set_value` command, the extra bits are not zero, and so an error is returned.

See Also

- current_time
- get_objects
- get_value
- report_values

setup_ip_static_library

(User-written application) Extract IP static files from the project or repository and prepare it for compile_simlib

Syntax

```
setup_ip_static_library [-directory <arg>] [-ip_repo_path <arg>]
    [-ips <arg>] [-library <arg>] [-project] [-install]
    [-no_update_catalog] [-force] [-quiet] [-verbose]
```

Returns

None

Usage

Name	Description
`[-directory]`	Extract static files in the specified directory Default: None
`[-ip_repo_path]`	Extract static files from the specified IP repository path Default: None
`[-ips]`	Extract static files for the specified IPs only Default: Empty
`[-library]`	Extract static files for the specified IP library Default: Empty
`[-project]`	Extract static files for the current project
`[-install]`	Extract static files for the IP catalog
`[-no_update_catalog]`	Do no update IP catalog Default: 1
`[-force]`	Overwrite static files
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution

Categories

xilinxtclstore

Description

Retrieve static simulation files for IP cores used in the current project, or from the Xilinx IP catalog, and create a source library for the `compile_simlib` command to use for compiling the IP files for a specified simulator.

Arguments

`-directory <arg>` - (Optional) Directory path to the static library. By default, if this option is not specified, then the library will be written to a directory named `static_compiled_lib` in the current working directory.

`-ip_repo_path args <args>` - (Optional) Extract static files from the specified IP repository paths.

`-ips <args>` - (Optional) Extract static files for the specified IP objects only. IP can be specified by the `get_ips` command.

`-library <args>` - (Optional) Extract static files for the specified IP library. IP libraries can be specified by name.

`-project` - (Optional) Extract and prepare static source library for the current project.

`-install <arg>` - (Optional) Extract and prepare static source library for the IP catalog.

`-no_update_catalog` - (Optional) Do no update IP catalog.

`-force` - (Optional) Overwrite existing files.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following command will build static library for all the IPs in the current project in `./static_compiled_lib`:

```
setup_ip_static_library -project
```

The following command will build static library for the current project in `/work/simlib`. The command will create the specified directory if it does not exist:

```
setup_ip_static_library -directory /work/simlib -project
```

See Also

- compile_simlib
- get_files
- get_ips

setup_pr_configurations

Creates minimum PR Configurations and Child Impl runs automatically based on the combination of Partition Instances and RMs

Syntax

```
setup_pr_configurations [-partitions <args>] [-use_netlist] [-force]
    [-run <arg>] [-quiet] [-verbose]
```

Usage

Name	Description
[-partitions]	List of partition instances and reconfig modules pairs
[-use_netlist]	Use netlist for getting instances of partition_defs to creating PR Configurations
[-force]	Using force deletes active parent impl run's PR Configuration and it's child runs and PR Configurations, and then creates new PR Configurations and runs
[-run]	Parent impl run to which child impl runs and PR Configurations need to be created
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Partition](#)

Description

Automatically creates the minimum PR configurations and child implementation runs based on the combination of Partition Instances and Reconfigurable Modules.

In the Partial Reconfiguration (PR) design flow, the PR configuration lets you specify a reconfigurable module (RM) to assign to a specific instance of a Partition Definition (partitionDef). This flow lets you create unique configurations of the design based on the combination of the core design and one or more RMs. The PR design flow requires the implementation of each PR configuration, resulting in partial bitstreams for the RMs, but complete bitstreams for each integrated configuration. Refer to the *Vivado Design Suite User Guide: Dynamic Function eXchange* (UG909) for more information.

This command is designed to work automatically to create the needed PR configurations and implementation runs for those configurations.

This command returns nothing if successful, or returns an error if the command fails.

Arguments

`-partitions <arg>` - (Optional) Specify a list of partition instances and reconfigurable module pairs to assign to the PR configuration. The argument must be specified as <partitionInstance>:<RM> pairs. This format lets you assign different RMs to multiple instances of a single partitionDef in the design.

`-force` - (Optional) Delete current PR configurations and configuration runs, and create new PR configurations and runs.

`-run <arg>` - (Optional) Specify the parent run to use for the implementation run of a PR configuration.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Example

The following example automatically creates the necessary PR configurations, and PR config implementation runs, to support the partitionDefs and RMs in the project:

```
setup_pr_configurations
```

See Also

- create_partition_def
- create_pr_configuration
- create_reconfig_module
- current_pr_configuration
- delete_pr_configurations
- get_pr_configurations

show_objects

Show objects in Find Results view

Syntax

```
show_objects [-name <arg>] [-quiet] [-verbose] <objects>
```

Usage

Name	Description
`[-name]`	Tab title
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<objects>`	Objects to show Find Results view

Categories

GUIControl

Description

Populates the specified objects into the Find Results window in the Vivado IDE.

Note: This command is only useful when run in the Vivado IDE. When run in Tcl or Batch mode the command simply returns without error or comment.

Arguments

`-name <arg>` - The name of the report tab to open in the Find Results window. If no name is specified, the default name of `find_1` is provided.

> **TIP:** *If the command is run multiple times, without the `-name` option, each occurrence of `show_objects` will overwrite the prior results.*

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<objects>` - The design objects to show in the Find Results window. The objects must be specified as design objects, and not just specified by name. Objects can be returned by commands like `get_cells`, `get_pins`, `get_nets`, or by `all_inputs`, and `all_rams`.

Examples

The following example shows all DSP objects in the current design in the Find Results window.

```
show_objects -name All_DSPs [all_dsps]
```

The following example shows all of the cells in the design hierarchy that are Clock or DSP PRIMITIVE_TYPEs:

```
show_objects -name find_1 [get_cells -hierarchical \
-filter { PRIMITIVE_TYPE =~ CLK.*.* || PRIMITIVE_TYPE =~ MULT.dsp.* } ]
```

See Also

- all_inputs
- all_rams
- get_cells
- get_nets
- get_pins
- get_ports

show_schematic

Show netlist items in schematic view

Syntax

```
show_schematic [-add] [-remove] [-regenerate] [-pin_pairs] [-name <arg>]
    [-quiet] [-verbose] <objects>
```

Usage

Name	Description
[-add]	Add to existing schematic view
[-remove]	Remove from existing schematic view
[-regenerate]	Regenerate layout of schematic view
[-pin_pairs]	objects are treated as pair of connected pins. This can be useful to display paths
[-name]	Schematic window title
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<objects>	Netlist items to show in schematic view

Categories

GUIControl

Description

Create a schematic view containing the specified design objects when the tool is invoked in GUI mode.

The scope of the schematic that is displayed depends on the objects specified. A schematic created with cells, shows the specified cells and any connections between the cells. A schematic created with pins, shows the pin objects, or shows them connected as appropriate if `-pin_pairs` is specified. A schematic created with nets shows the nets, as well as the cells and ports connected to the nets.

To display a schematic with multiple levels of hierarchy, use the `current_instance` command to set the top-level of the hierarchy, or the level of interest, and use the `-hierarchical` option when specifying design objects with a `get_*` command.

Note: This command is only useful when run in the Vivado IDE. When run in Tcl or Batch mode the command simply returns without error or comment.

Send Feedback

Arguments

`-add` - (Optional) Add the specified objects to the schematic window.

`-remove` - (Optional) Remove the specified objects from the schematic window.

`-regenerate` - (Optional) Regenerate the schematic window.

`-pin_pairs` - (Optional) When specified with a pair of connected pin objects, the schematic shows the pins and the wire between the pins. When the `-pin_pairs` option is not specified, or is specified with disconnected pins, the wire is not shown.

`-name <arg>` - (Optional) Defines a name for the schematic window opened in the GUI. Use this name to add to, remove from, or regenerate the schematic.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<objects>` - (Required) The netlist objects to display in the schematic window.

Examples

The following example creates a schematic for the top-level of the design, displaying the nets as well as the ports and cells they connect to:

```
show_schematic [get_nets]
```

The following example sets the level of hierarchy of interest, and creates a hierarchical schematic from the current level down:

```
current_instance A
show_schematic [get_nets -hier]
```

The following example creates a schematic window showing the specified pins, and the wire connection between them:

```
show_schematic -pin_pairs [get_pins {data0_i/O data_reg/D}]
```

See Also

- current_instance

- get_cells
- get_nets
- get_pins
- get_ports

split_diff_pair_ports

Remove differential pair relationship between 2 ports

Syntax

```
split_diff_pair_ports [-quiet] [-verbose] <ports>...
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<ports>	Ports to split

Categories

PinPlanning

Description

Splits an existing differential pair of ports into two single-ended ports.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<ports> - (Required) The names of two ports of a differential pair to split into single-ended ports.

Examples

The following example splits the specified diff pair ports to form two single ended ports:

```
split_diff_pair_ports PORT_N PORT_P
```

Send Feedback

See Also

- make_diff_pair_ports
- create_port
- create_interface

Send Feedback

start_gui

Start GUI

Syntax

```
start_gui [-verbose]
```

Usage

Name	Description
[-verbose]	Suspend message limits during command execution

Categories

GUIControl

Description

Launches the GUI when the tool is running in the Vivado Design Suite Tcl shell. The GUI is invoked with the current project, design, and run information.

Arguments

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example is executed from the command prompt when the tool is running in Tcl mode:

```
Vivado% start_gui
```

See Also

- stop_gui

start_vcd

Start capturing VCD output (equivalent of $dumpon verilog system task). This can be used after a stop_vcd TCL command has stopped VCD generation started by open_vcd

Syntax

```
start_vcd [-quiet] [-verbose]
```

Usage

Name	Description
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution

Categories

Simulation

Description

The `start_vcd` command specifies that the tool start writing Value Change Dump (VCD) information into the specified VCD object. This Tcl command models the behavior of the Verilog `$dumpon` system task.

> **IMPORTANT!** *You must execute the `open_vcd` command before using the `start_vcd` command.*

Nothing is returned by this command.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Send Feedback

Examples

The following example starts the writing of HDL signals into the current VCD file:

```
start_vcd
```

See Also

- close_vcd
- open_vcd
- stop_vcd

Send Feedback

startgroup

Start a set of commands that can be undone/redone as a group

Syntax

```
startgroup [-try] [-quiet] [-verbose]
```

Returns

Int

Usage

Name	Description
`[-try]`	Don't start a group if one has already been started
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution

Categories

GUIControl

Description

Starts a sequence of commands that can be undone or redone as a series. Use `endgroup` to end the sequence of commands.

You can have multiple command groups to `undo` or `redo`, but you cannot nest command groups. You must use `endgroup` to end a command sequence before using `startgroup` to create a new command sequence.

> 💡 **TIP:** *The `startgroup/endgroup` commands are provided to support sequences of related commands that can be undone via the `undo` command, or redone if needed using the `redo` command. However, some commands can trigger an `endgroup` unexpectedly, and certain commands do not support either `undo` or `redo`. The limitations are not fully defined.*

The `startgroup` command returns an integer value of 0 if a group is already started, and returns an integer value of 1 if the startgroup command has started a new group.

Arguments

`-try` - (Optional) Returns 1 if a new group is started. Returns 0 if a group has already been started, and therefore a new group cannot be started.

Send Feedback

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example defines a `startgroup`, executes a sequence of related commands, and then executes the `endgroup`. This sequence of commands can be undone or redone as a group:

```
startgroup
create_pblock pblock_wbArbEngine
create_pblock pblock_usbEngnSRM
add_cells_to_pblock pblock_wbArbEngine \
    [get_cells [list wbArbEngine]] -clear_locs
add_cells_to_pblock pblock_usbEngnSRM \
    [get_cells [list usbEngine1/usbEngineSRAM]] -clear_locs
endgroup
```

See Also

- endgroup
- redo
- undo

step

Step simulation to the next statement

Syntax

```
step [-quiet] [-verbose]
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Simulation

Description

Step the current simulation to the next executable statement in the HDL source files.

The line stepping feature lets you run the simulator stepping through the source code line-by-line. This is helpful if you are interested in observing how each line or feature of your HDL source affects the results of simulation.

The step command returns information related to the next executable line from the HDL source file, or returns an error if it fails.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the set_msg_config command.

Examples

The following example runs the current executable line of the HDL source code, and pauses at the next executable line, returning information about that line:

```
step
Stopped at time : 0 fs : File "C:/Data/ug937/sim/testbench.v" Line 17
```

See Also

- run
- stop

Send Feedback

stop

Use within a condition to tell simulation to stop when a condition is true

Syntax

```
stop [-quiet] [-verbose]
```

Returns

A "stop" in simulation is a pause and not a quit.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Simulation

Description

The `stop` command pauses the current simulation. This command can be used within a condition, defined by `add_condition`, to pause the simulation when the condition is true.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example defines a condition named resetLow, that becomes true when the reset signal is low, and then puts a message to the standard output, and stops the current simulation:

```
add_condition -name resetLow {/testbench/reset == 0 } {
puts "Condition Reset was encountered at [current_time]. \
    Stopping simulation."
stop }
```

See Also

- add_condition
- report_conditions
- restart
- run
- step

stop_gui

Close GUI

Syntax

```
stop_gui [-verbose]
```

Usage

Name	Description
[-verbose]	Suspend message limits during command execution

Categories

GUIControl

Description

Stops GUI mode and places the tool into Tcl mode, running in the Vivado Design Suite Tcl shell. In Tcl mode, all commands must be entered as Tcl commands or through Tcl scripts.

Arguments

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example stops and closes the GUI and places the tool into Tcl mode:

```
stop_gui
```

See Also

- start_gui

Send Feedback

stop_hw_hbm_amon

Disable Activity Monitor runs for the specified hardware HBM(s)

Syntax

```
stop_hw_hbm_amon [-quiet] [-verbose] <hw_objects>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hw_objects>	hardware objects

Categories

Hardware

Description

The `stop_hw_hbm_amon` command stops a running HBM activity monitor in the Vivado hardware manager that has been previously started using the `run_hw_hbm_amon` command.

This command returns nothing when successful, or returns an error when it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_objects> - (Required) Defines the HBM activity monitor objects (hw_hbms) to stop.

Examples

The following example stops the HBM activity monitor for the associated HBM core:

```
stop_hw_hbm_amon [get_hw_hbms *HBM_2]
```

See Also

- add_hw_hbm_pc
- commit_hw_hbm
- current_hw_device
- current_hw_target
- get_hw_hbms
- refresh_hw_hbm
- remove_hw_hbm_pc
- resume_hw_hbm_amon
- run_hw_hbm_amon

stop_hw_sio_scan

Stop hardware SIO scans.

Syntax

```
stop_hw_sio_scan [-quiet] [-verbose] <hw_sio_scans>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hw_sio_scans>	hardware SIO scans

Categories

Hardware

Description

Stop the specified scan running in the Vivado serial I/O analyzer.

To analyze the margin of a given link, it is often helpful to run a scan of the link using the specialized Eye Scan hardware of Xilinx UltraScale devices or 7 series FPGAs. The Vivado serial I/O analyzer feature lets you to create, run, and save link scans.

This command lets you stop a scan that is in progress as started with the `run_hw_sio_scan` command.

You can remove the created scan object using `remove_hw_sio_scan`.

This command returns a message if successful, or returns an error if the command fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Send Feedback

<hw_sio_scans> - (Required) Specify one or more hw_sio_scan objects to stop. The hw_sio_scan must be specified as an object as returned by the `get_hw_sio_scans` commands.

Example

The following example launches a scan using the run_hw_sio_scan command, and then stops the specified scan:

```
run_hw_sio_scan [lindex [get_hw_sio_scans {SCAN_3}] 0]
stop_hw_sio_scan [get_hw_sio_scans SCAN_3]
```

See Also

- create_hw_sio_scan
- current_hw_device
- get_hw_sio_scans
- remove_hw_sio_scan
- run_hw_sio_scan
- wait_on_hw_sio_scan
- write_hw_sio_scan

stop_hw_sio_sweep

Stop hardware SIO sweeps.

Syntax

```
stop_hw_sio_sweep [-quiet] [-verbose] <hw_sio_sweeps>
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hw_sio_sweeps>	hardware SIO sweeps

Categories

Hardware

Description

Stop the specified sweep scan.

To analyze the margin of a given link, it is often helpful to run a scan of the link using the specialized features of Xilinx UltraScale devices or 7 series FPGAs. It can also be helpful to run multiple scans on a the link with different configuration settings for the GTs. This can help you determine which settings are best for your design. The Vivado serial I/O analyzer feature enables you to define, run, and save link sweeps, or collections of link scans run across a range of values.

This command lets you stop a sweep scan that is in progress as started with the run_hw_sio_sweep command.

You can remove the created sweep scan object using remove_hw_sio_sweep.

This command returns nothing if successful, or returns an error if the command fails.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

Send Feedback

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_sio_sweeps> - (Required) Specify one or more hw_sio_sweep objects to stop. The hw_sio_sweep must be specified as an object as returned by the `get_hw_sio_sweeps` command.

Example

The following example stops the specified running sweep scan:

```
stop_hw_sio_sweep [lindex [get_hw_sio_sweeps {SWEEP_0}] 0]
```

See Also

- create_hw_sio_sweep
- current_hw_device
- get_hw_sio_sweeps
- remove_hw_sio_sweep
- run_hw_sio_sweep
- wait_on_hw_sio_sweep
- write_hw_sio_sweep

Send Feedback

stop_vcd

Stop capturing VCD output (equivalent of $dumpoff verilog system task). The start_vcd TCL command can be used to resume capturing VCD

Syntax

```
stop_vcd [-quiet] [-verbose]
```

Usage

Name	Description
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution

Categories

Simulation

Description

Stop writing the simulation values to the current Value Change Dump (VCD) file. This suspends the output of simulation information to the file until the process is resumed using the `start_vcd` command.

This Tcl command models the behavior of the Verilog `$dumpoff` system task.

> **IMPORTANT!** *You must execute the `open_vcd` command before using the `stop_vcd` command.*

Nothing is returned by the command.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Send Feedback

Examples

The following example stops writing simulation values to the current VCD file:

```
stop_vcd
```

See Also

- close_vcd
- open_vcd
- start_vcd

Send Feedback

swap_locs

Swap two locations

Syntax

```
swap_locs [-quiet] [-verbose] <aloc> <bloc>
```

Usage

Name	Description
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<aloc>`	First location (port/cell/site - should be of same type as 'bloc')
`<bloc>`	Second location (port/cell/site - should be of same type as 'aloc')

Categories

[Floorplan](#)

Description

Swaps the LOC constraints assigned to two similar logic elements. A logic element is an element that can be placed onto a device resource on the FPGA.

Some DRC checking is performed when the `swap_locs` command is executed to ensure that the two selected elements can in fact be assigned to their new locations. If the location of either element is invalid for any reason, the `swap_locs` command will fail and an error will be returned.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<aloc> - (Required) The location of the first logic element to swap. This can be specified as a port, a cell, or a device site.

<bloc> - (Required) The location of the second logic element to swap. This can be specified as a port, a cell, or a device site. This must match the type specified by the <aloc> variable.

Examples

The following example swaps the instances assigned to the two specified device sites:

```
swap_locs [get_sites {OLOGIC_X2Y1}] [get_sites {OLOGIC_X2Y0}]
```

See Also

- get_cells
- get_ports
- get_sites

Send Feedback

synth_design

Synthesize a design using Vivado Synthesis and open that design

Syntax

```
synth_design [-name <arg>] [-part <arg>] [-constrset <arg>] [-top <arg>]
    [-include_dirs <args>] [-generic <args>] [-verilog_define <args>]
    [-flatten_hierarchy <arg>] [-gated_clock_conversion <arg>]
    [-directive <arg>] [-rtl] [-bufg <arg>] [-no_lc]
    [-shreg_min_size <arg>] [-mode <arg>] [-fsm_extraction <arg>]
    [-rtl_skip_mlo] [-rtl_skip_ip] [-rtl_skip_constraints]
    [-srl_style <arg>] [-keep_equivalent_registers]
    [-resource_sharing <arg>] [-cascade_dsp <arg>]
    [-control_set_opt_threshold <arg>] [-incremental <arg>]
    [-max_bram <arg>] [-max_uram <arg>] [-max_dsp <arg>]
    [-max_bram_cascade_height <arg>] [-max_uram_cascade_height <arg>]
    [-retiming] [-no_srlextract] [-assert] [-no_timing_driven] [-sfcu]
    [-debug_log] [-quiet] [-verbose]
```

Returns

Design object

Usage

Name	Description
[-name]	Design name
[-part]	Target part
[-constrset]	Constraint fileset to use
[-top]	Specify the top module name
[-include_dirs]	Specify verilog search directories
[-generic]	Specify generic parameters. Syntax: -generic <name>=<value> -generic <name>=<value> ...
[-verilog_define]	Specify verilog defines. Syntax: -verilog_define <macro_name>[=<macro_text>] -verilog_define <macro_name>[=<macro_text>] ...
[-flatten_hierarchy]	Flatten hierarchy during LUT mapping. Values: full, none, rebuilt Default: rebuilt
[-gated_clock_conversion]	Convert clock gating logic to flop enable. Values: off, on, auto Default: off
[-directive]	Synthesis directive. Values: default, RuntimeOptimized, AreaOptimized_high, AreaOptimized_medium,AlternateRoutability, AreaMapLargeShiftRegToBRAM, AreaMultThresholdDSP, FewerCarryChains,Perfor manceOptimized, LogicCompaction Default: default
[-rtl]	Elaborate and open an rtl design
[-bufg]	Max number of global clock buffers used by synthesis Default: 12

Name	Description
[-no_lc]	Disable LUT combining. Do not allow combining LUT pairs into single dual output LUTs.
[-shreg_min_size]	Minimum length for chain of registers to be mapped onto SRL Default: 3
[-mode]	The design mode. Values: default, out_of_context Default: default
[-fsm_extraction]	FSM Extraction Encoding. Values: off, one_hot, sequential, johnson, gray, user_encoding, auto Default: auto
[-rtl_skip_mlo]	Skip mandatory logic optimization for RTL elaboration of the design; requires -rtl option.
[-rtl_skip_ip]	Exclude subdesign checkpoints in the RTL elaboration of the design; requires -rtl option.
[-rtl_skip_constraints]	Do not load and validate constraints against elaborated design; requires -rtl option.
[-srl_style]	Static SRL Implementation Style. Values: register, srl, srl_reg, reg_srl, reg_srl_reg
[-keep_equivalent_registers]	Prevents registers sourced by the same logic from being merged. (Note that the merging can otherwise be prevented using the synthesis KEEP attribute)
[-resource_sharing]	Sharing arithmetic operators. Value: auto, on, off Default: auto
[-cascade_dsp]	Controls how adders summing DSP block outputs will be implemented. Value: auto, tree, force Default: auto
[-control_set_opt_threshold]	Threshold for synchronous control set optimization to lower number of control sets. Valid values are 'auto' and non-negative integers. The higher the number, the more control set optimization will be performed and fewer control sets will result. To disable control set optimization completely, set to 0. Default: auto
[-incremental]	Incremental synthesis directive with different strategies targeting at runtime or qor. Value of this is the strategy name.
[-max_bram]	Maximum number of block RAM allowed in design. (Note -1 means that the tool will choose the max number allowed for the part in question) Default: -1
[-max_uram]	Maximum number of Ultra RAM blocks allowed in design. (Note -1 means that the tool will choose the max number allowed for the part in question) Default: -1
[-max_dsp]	Maximum number of block DSP allowed in design. (Note -1 means that the tool will choose the max number allowed for the part in question) Default: -1
[-max_bram_cascade_height]	Controls the maximum number of BRAM that can be cascaded by the tool. (Note -1 means that the tool will choose the max number allowed for the part in question) Default: -1
[-max_uram_cascade_height]	Controls the maximum number of URAM that can be cascaded by the tool. (Note -1 means that the tool will choose the max number allowed for the part in question) Default: -1
[-retiming]	Seeks to improve circuit performance for intra-clock sequential paths by automatically moving registers (register balancing) across combinatorial gates or LUTs. It maintains the original behavior and latency of the circuit and does not require changes to the RTL sources.
[-no_srlextract]	Prevents the extraction of shift registers so that they get implemented as simple registers

Send Feedback

Name	Description
`[-assert]`	Enable VHDL assert statements to be evaluated. A severity level of failure will stop the synthesis flow and produce an error.
`[-no_timing_driven]`	Do not run in timing driven mode
`[-sfcu]`	Run in single-file compilation unit mode
`[-debug_log]`	Print detailed log files for debugging
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution

Categories

[Tools](#)

Description

Directly launches the Vivado synthesis engine to compile and synthesize a design in either Project Mode or Non-Project Mode in the Vivado Design Suite. Refer to the *Vivado Design Suite User Guide: Design Flows Overview (UG892)* for a complete description of Project Mode and Non-Project Mode.

Vivado synthesis can be launched directly with the `synth_design` command in the Non-Project Mode of the Vivado Design Suite.

> **TIP:** *The `synth_design` can be multi-threaded to speed the process. Refer to the `set_param` command for more information on setting the `general.maxThreads` parameter.*

In Project Mode, synthesis should be launched from an existing synthesis run created with the `create_run` command. The run is launched using the `launch_runs` command, and this in turn calls `synth_design` for Vivado synthesis.

You can also use the `synth_design` command to elaborate RTL source files, and open an elaborated design:

```
synth_design -rtl -name rtl_1
```

This command returns a transcript of the synthesis process, or returns an error if it fails.

Arguments

`-name <arg>` - (Optional) This is the name assigned to the synthesized design when it is opened by the Vivado tool after synthesis has completed. This name is for reference purposes, and has nothing to do with the top-level of the design or any logic contained within.

`-part <arg>` - (Optional) The target Xilinx device to use for the design. If the part is not specified the default part assigned to the project will be used.

`-constrset <arg>` - (Optional) The name of the XDC constraints to use when synthesizing the design. Vivado synthesis requires the use of XDC, and does not support UCF. The `-constrset` argument must refer to a constraint fileset that exists. It cannot be used to create a new fileset. Use the `create_fileset` command for that purpose.

`-top <arg>` - (Optional) The top module of the design hierarchy.

> ⭐ **IMPORTANT!** *If you use the `find_top` command to define the `-top` option, be sure to specify only one top if `find_top` returns multiple prospects. See the examples below.*

`-include_dirs <args>` - (Optional) The directories to search for Verilog `` `include `` files. You can specify multiple directories by creating a list to contain them:

```
-include_dirs {C:/data/include1 C:/data/include2}
```

`-generic <name>=<value>` - (Optional) The value of a VHDL generic entity, or of a Verilog parameter. The `-generic` option can be used to override the assigned values of parameters in the RTL design sources. However it can only override parameters at the top level of the design. The parameters of lower-level modules can only be overridden at the time of instantiation and not by the `-generic` option. The syntax for the `-generic` argument is `<name>=<value>`, specifying the name of the generic or parameter, and the value to be assigned. Repeat the `-generic` option multiple times in the `synth_design` command for each generic or parameter value to be defined:

```
synth_design -generic width=32 -generic depth=512 ...
```

> ⭐ **IMPORTANT!** *When specifying binary values for boolean or std_logic VHDL generic types, you must specify the value using the Verilog bit format, rather than standard VHDL format:*
>
> ```
> 0 = 1'b0
> 01010000 = 8'b01010000
> ```

`-verilog_define <name>=<text>` - (Optional) Set values for Verilog `` `define ``, and `` `ifdef ``, statements. The syntax for the `-verilog_define` argument is `<name>=<text>`, specifying the name of the define directive, and the value to be assigned. The argument can be reused multiple times in a single `synth_design` command.

```
synth_design -verilog_define <name>=<value> -verilog_define
<name>=<value> ...
```

`-flatten_hierarchy <arg>` - (Optional) Flatten the hierarchy of the design during LUT mapping. The valid values are:

- `rebuilt` - This will attempt to rebuild the original hierarchy of the RTL design after synthesis has completed. This is the default setting.

- `full` - Flatten the hierarchy of the design.

Send Feedback

- `none` - Do not flatten the hierarchy of the design. This will preserve the hierarchy of the design, but will also limit the design optimization that can be performed by the synthesis tool.

`-gated_clock_conversion <arg>` - (Optional) Convert clock gating logic to use the flop enable pins when available. This optimization can eliminate logic and simplify the netlist. Refer to the `GATED_CLOCK` property in the *Vivado Design Suite User Guide: Synthesis (UG901)* for more information. Valid values for this option are:

- `off` - Disables the conversion of clock gating logic during synthesis, regardless of the use of the `GATED_CLOCK` property in the RTL design.

- `on` - Converts clock gating logic based on the use of the `GATED_CLOCK` property in the RTL design.

- `auto` - lets Vivado synthesis perform gated clock conversion if either the `GATED_CLOCK` property is present in the RTL, or if the Vivado tool detects a gate with a valid clock constraint.

`-directive <arg>` - (Optional) Direct synthesis to achieve specific design objectives. Only one directive can be specified for a single `synth_design` command, and values are case-sensitive. Valid values are:

- `default` - Run the default synthesis process.

- `runtimeoptimized` - Perform fewer timing optimizations and eliminate some RTL optimizations to reduce synthesis run time.

- `AreaOptimized_high` - Perform general area optimizations including AreaMapLargeShiftRegToBRAM, AreaThresholdUseDSP directives.

- `AreaOptimized_medium` - Perform general area optimizations including forcing ternary adder implementation, applying new thresholds for use of carry chain in comparators, and implementing area optimized multiplexers.

- `AlternateRoutability` - Algorithms to improve routability with reduced use of MUXFs and CARRYs.

- `AreaMapLargeShiftRegToBRAM` - Detects large shift registers and implements them using dedicated blocks of RAM.

- `AreaMultThresholdDSP` - Lower threshold for dedicated DSP block inference for packing multipliers.

- `FewerCarryChains` - Higher operand size threshold to use LUTs instead of the carry chain.

- `PerformanceOptimized` - Perform general timing optimizations including logic level reduction at the expense of area.

`-rtl` - (Optional) Elaborate the HDL source files and open the RTL design. In designs that use out-of-context (OOC) modules, such as IP from the Xilinx IP catalog, the Vivado Design Suite will import synthesized design checkpoints (DCP) for the OOC modules in the design, and import associated constraint files (XDC) into the elaborated design. However, you can disable the default behavior using the `-rtl_skip_ip` and `-rtl_skip_constraints` options.

`-rtl_skip_ip` - (Optional) This option requires the use of the `-rtl` option. When elaborating the RTL design, this option causes the Vivado Design Suite to skip loading the DCP files for OOC modules in the design, and instead load a stub file to treat the OOC modules as black boxes. This can significantly speed elaboration of the design.

> 💡 **TIP:** *An OOC synthesis run will be needed in either case to generate the DCP file that is loaded during elaboration, or to generate the stub file needed for the black box.*

`-rtl_skip_constraints` - (Optional) This option requires the use of the `-rtl` option. When elaborating the RTL design, this option causes the Vivado Design Suite to skip loading any design constraints (XDC) into the elaborated design. This can significantly speed elaboration of the design.

`-bufg <arg>` - (Optional) Specify the maximum number of global clock buffers to be used on clock nets during synthesis. Specified as a value >= 1, which should not exceed the BUFG count on the target device. The default value is 12.

> 💡 **TIP:** *Vivado synthesis infers up to the number of BUFGs specified, including those instantiated in the RTL source. For example, with the default setting of `-bufg 12`, if there are three BUFGs instantiated in the RTL, the tool infers up to nine more for a total of 12.*

`-no_lc` - (Optional) Disable the default LUT combining feature of Vivado synthesis.

`-shreg_min_size <arg>` - (Optional) Specified as an integer, this is the minimum length for a chain of registers to be mapped onto SRL. The default is three.

`-mode [ default | out_of_context ]` - (Optional) Out of Context mode specifies the synthesis of an IP module, or block module, for use in an out-of-context design flow. This mode turns off I/O buffer insertion for the module, and marks the module as OOC, to facilitate its use in the tool flow. The block can also be implemented for analysis purposes. Refer to the *Vivado Design Suite User Guide: Designing with IP (UG896)* or the *Vivado Design Suite User Guide: Hierarchical Design (UG905)* for more information.

`-fsm_extraction <arg>` - (Optional) Finite state machine (FSM) encoding is automatic (`auto`) in Vivado synthesis by default. This option enables state machine identification and specifies the type of encoding that should be applied. Valid values are: `off`, `one_hot`, `sequential`, `johnson`, `gray`, `auto`. Automatic encoding (`auto`) allows the tool to choose the best encoding for each state machine identified. In this case, the tool may use different encoding styles for different FSMs in the same design.

Note: Use `-fsm_extraction off` to disable finite state machine extraction in Vivado synthesis. This will override the FSM_ENCODING property when specified.

`-keep_equivalent_registers` - (Optional) Works like the KEEP property to prevent the merging of registers during optimization.

`-resource_sharing <arg>` - (Optional) Share arithmetic operators like adders or subtractors between different signals, rather than creating new operators. This can result in better area usage when it is turned on. Valid values are: auto, on, off. The default is auto.

`-cascade_dsp [ auto | tree | force ]` - (Optional) Specifies how to implement adders that add DSP block outputs. Valid values include `auto`, `tree`, `force`. The default setting is `auto`.

`-control_set_opt_threshold <arg>` - (Optional) Threshold for synchronous control set optimization to decrease the number of control sets. Specifies how large the fanout of a control set should be before it starts using it as a control set. For example, if `-control_set_opt_threshold` is set to 10, a synchronous reset that only fans out to 5 registers would be moved to the D input logic, rather than using the reset line of a register. However, if `-control_set_opt_threshold` is set to 4, then the reset line is used. This option can be specified as "`auto`", or as an integer from 0 to 16. The default setting is "`auto`", and the actual threshold used under "auto" can vary depending on the selected device architecture.

`-incremental` - (Optional) Specify a DCP file for the incremental compilation flow. In the incremental synthesis flow, the netlist from the incremental DCP is applied to the design objects in the current design to reuse existing synthesis results when possible.

`-max_bram <arg>` - (Optional) Specify the maximum number of Block RAM to add to the design during synthesis. Specify a value >= 1, which should not exceed the available BRAM count on the target device. If a value of -1 is used, the Vivado synthesis tool will not exceed the available Block RAM limit of the device. The default value is -1.

Note: A value of 0 directs Vivado synthesis to not infer BRAMs in the design, but is not a recommended value.

`-max_uram <arg>` - (Optional) Specify the maximum number of Ultra RAM blocks (URAM) to add to the design during synthesis. Specify a value >= 1, which should not exceed the available URAM count on the target device. If a value of -1 is used, the Vivado synthesis tool will not exceed the available URAM block limit of the device. The default value is -1.

Note: A value of 0 directs Vivado synthesis to not infer URAM in the design, but is not a recommended value.

`-max_dsp <arg>` - (Optional) Specify the maximum number of DSPs to add to the design during synthesis. Specify a value >= 1, which should not exceed the available DSP count on the target device. If a value of -1 is used, the Vivado synthesis tool will not exceed the available limit of the device. The default value is -1.

Note: A value of 0 directs Vivado synthesis to not infer DSPs in the design, but is not a recommended value.

`-max_bram_cascade_height <arg>` - (Optional) Controls the maximum number of BRAM that can be cascaded by the tool. A value of -1 lets Vivado synthesis choose up to the maximum number allowed for the target part. The default value is -1.

`-max_uram_cascade_height <arg>` - (Optional) Controls the maximum number of URAM that can be cascaded by the tool. A value of -1 lets Vivado synthesis choose up to the maximum number allowed for the target part. The default value is -1.

`-retiming` - (Optional) Seeks to improve circuit performance for intra-clock sequential paths by automatically moving registers (register balancing) across combinatorial gates or LUTs. It maintains the original behavior and latency of the circuit and does not require changes to the RTL sources.

`-no_srlextract` - (Optional) Prevents the extraction of shift registers so that they get implemented as simple registers.

`-assert` - (Optional) Enable VHDL assert statements to be evaluated. A severity level of failure will stop the synthesis flow and produce an error.

`-no_timing_driven` - (Optional) Disables the default timing driven synthesis algorithm. This results in a reduced synthesis runtime, but ignores the effect of timing on synthesis.

`-debug_log` - (Optional) Prints an extended log file that is useful for debugging. Will have extra info about logic that was trimmed along with sections on the sizes of the RTL partitions and RTL Components. In addition there will be more INFO/WARNING messages that are often ignored.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example uses the `set_property` command to define the target part for the active project, then elaborates the source files and opens an RTL design:

```
set_property part xc7vx485tffg1158-1 [current_project]
synth_design -rtl -name rtl_1
```

Note: The default source set, constraint set, and part will be used in this example.

The following example uses the `find_top` command to define the top of the current design for synthesis:

```
synth_design -top [lindex [find_top] 0]
```

Note: Since `find_top` returns multiple possible candidates, choosing index 0 chooses the best top candidate for synthesis.

The following example runs synthesis on the current design, defining the top module and the target part, and specifying no flattening of the hierarchy. The results of the synthesis run are then opened in a netlist design:

```
synth_design -top top -part xc7k70tfbg676-2 -flatten_hierarchy none
open_run synth_1 -name netlist_1
```

See Also

- create_ip_run
- create_run
- current_design
- current_project
- find_top
- open_run
- opt_design
- set_property

synth_ip

Generate a synthesis netlist for an IP

Syntax

```
synth_ip [-force] [-quiet] [-verbose] <objects>
```

Usage

Name	Description
`[-force]`	Force regeneration of the netlist.
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<objects>`	All the objects for which a netlist needs to be generated for.

Categories

[Project](), [IPFlow]()

Description

This command is used in the non-project flow to create a synthesized design checkpoint file (DCP) to support the out-of-context (OOC) IP flow, or to synthesize and implement an IP module in the OOC hierarchical design flow. IP objects are specified by the `get_ips` command, or for the specified IP core file (XCI) as specified by the `get_files` command.

> ⭐ **IMPORTANT!** *To enable this functionality, the IP core must be marked for OOC generation by setting the GENERATE_SYNTH_CHECKPOINT property to true (or 1) using the `set_property` command on the XCI file.*

For project-based designs you would use the `create_ip_run` and `launch_runs` commands. Refer to the *Vivado Design Suite User Guide: Design Flows Overview* (UG892) for more information on Project and Non-Project Modes in Vivado.

The `synth_ip` command will automatically generate any required target files prior to synthesizing the IP core. The source files required to synthesize the IP are copied into the IP run directory. Upon completion, any newly generated OOC target files (dcp, stub files, funcsim netlists...) are registered with the associated IP core.

Arguments

`-force` - (Optional) Force re-synthesis of the specified IP objects, even if the generated output products for the specified IP are all current.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<objects>` - (Required) One or more IP objects to synthesize. The objects can be specified as IP cores by the `get_ips` command, or as XCI files using the `get_files` command.

Examples

The following example synthesizes the specified IP object, regenerating the netlist if the synthesized core is up-to-date:

```
synth_ip [get_ips char_fifo] -force
```

See Also

- create_ip_run
- generate_target
- get_files
- get_ips
- launch_runs
- synth_design
- read_checkpoint

Send Feedback

tie_unused_pins

Tie off unused cell pins

Syntax

```
tie_unused_pins [-of_objects <args>] [-quiet] [-verbose]
```

Usage

Name	Description
[-of_objects]	tie unused pins of specified cell(s)
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Netlist

Description

Tie up or down the unconnected pins of cells in the open synthesized or implemented design. The command uses an internal process to identify whether a pin should be tied up or down.

This command is intended to tie up or down the unconnected pins of cells added to the netlist with the `create_cell` command.

Arguments

`-of_objects` <args> - (Optional) Tie up or down the unused pins of the specified cell objects.

Note: The `-of_objects` option requires objects to be specified using the `get_*` commands, such as `get_cells` or `get_pins`, rather than specifying objects by name. In addition, `-of_objects` cannot be used with a search <pattern>.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Example

The following example ties the unused pins of cells up or down, depending on their usage:

```
tie_unused_pins -of_objects [get_cells cpuEngine]
```

See Also

- create_cell
- create_pin
- get_cells
- get_pins

undo

Un-do previous command

Syntax

```
undo [-list] [-quiet] [-verbose]
```

Returns

With -list, the list of undoable tasks.

Usage

Name	Description
[-list]	Show a list of undoable tasks
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

GUIControl

Description

> ⭐ **IMPORTANT!** *The `undo` and `redo` commands are intended for use in the Vivado IDE, and are not recommended for use in Tcl scripts to restore designs to a former state. To restore a design to a specific condition, you must write a design checkpoint using the `write_checkpoint` command, to be restored using `read_checkpoint`.*

Undo a prior command. This command can be used repeatedly to undo a series of commands.

If a group of commands has been created using the `startgroup` and `endgroup` commands, this command will undo that group as a sequence. The `undo` command will start at the `endgroup` command and continue to undo until it hits the `startgroup` command.

If you `undo` a command, and then change your mind, you can `redo` the command.

Arguments

`-list` - (Optional) Reports the list of commands that can be undone. As you execute the undo command, the tool will step backward through this list of commands.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example returns a list of commands that you can undo:

```
undo -list
```

See Also

- redo
- startgroup
- endgroup

ungroup_bd_cells

Move the group of cells inside a hierarchy cell to its parent cell, and then remove the hierarchical cell. The connections between these cells are maintained; the connections between these cells and other cells are maintained through crossing hierarchy cell.

Syntax

```
ungroup_bd_cells [-prefix <arg>] [-quiet] [-verbose] [<cells>...]
```

Returns

0 if success.

Usage

Name	Description
[-prefix]	Prefix name to add to cells
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<cells>]	Match engine names against cell names Default: *

Categories

IPIntegrator

Description

This command is intended to undo the grouping of IP integrator cells into a hierarchical module, by either the `group_bd_cells` or the `move_bd_cells` commands. The command moves the cells inside a selected hierarchical module up one level to the parent cell, and then removes the hierarchical module.

The connections between the selected cells are maintained. The connections between these cells and other cells are maintained automatically by removing unneeded subsystem ports and pins.

This command returns 0 if successful, or an error message if it fails.

Arguments

`-prefix <arg>` - (Optional) A prefix name to apply to any cells that are moved up one level in the hierarchy.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<cells>` - (Required) A hierarchical module defined by the `group_bd_cells` or `move_bd_cells` commands. Only one hierarchical module can be specified for the command at one time.

Example

The following example:

```
ungroup_bd_cells -prefix up2_ [get_bd_cells myMod2]
```

See Also

- create_bd_cell
- get_bd_cells
- group_bd_cells
- move_bd_cells

unhighlight_objects

Unhighlight objects that are currently highlighted

Syntax

```
unhighlight_objects [-color_index <arg>] [-rgb <args>] [-color <arg>]
    [-leaf_cells] [-quiet] [-verbose] [<objects>]
```

Usage

Name	Description
`[-color_index]`	Color index
`[-rgb]`	RGB color index list
`[-color]`	Valid values are red green blue magenta yellow cyan and orange
`[-leaf_cells]`	Leaf cells
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`[<objects>]`	Objects to unhighlight

Categories

GUIControl

Description

This command is for use in GUI mode. This command unhighlights the specified object or objects that were previously highlighted by the `highlight_objects` command.

This command supports the color options as specified below. These options can be used to unhighlight all objects currently highlighted in the specified color. See the example below.

Arguments

`-color_index <arg>` - (Optional) Valid value is an integer from 1 to 20. Specify the color index to unhighlight. The color index is defined by the **Colors → Highlight** category of the **Tools → Settings** dialog box. Refer to the *Vivado Design Suite User Guide: Using the IDE (UG893)* for more information on setting themes.

`-rgb <args>` - (Optional) Specify the color to unhighlight in the form of an RGB code specified as {R G B}. For instance, {255 255 0} specifies the color yellow.

`-color <arg>` - (Optional) Specify the color to unhighlight by name. Supported highlight colors are red, green, blue, magenta, yellow, cyan, and orange.

Send Feedback

`-leaf_cells <arg>` - (Optional) Specifies one or more cell names or objects to unhighlight based on the specified color.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<objects>` - (Optional) Specifies one or more objects to be unhighlighted. If no objects are specified, all highlighted objects of the specified color will be unhighlighted. If no color is specified, all highlighted objects will be unhighlighted.

Examples

The following example unhighlights the specified cell objects:

```
unhighlight_objects -leaf_cells [get_cells cpuEngine/*]
```

The following example unhighlights all objects currently highlighted in the color yellow:

```
unhighlight_objects -color yellow
```

See Also

- get_selected_objects
- highlight_objects

unmark_objects

Unmark items that are currently marked

Syntax

```
unmark_objects [-rgb <args>] [-color <arg>] [-quiet] [-verbose] [<objects>]
```

Usage

Name	Description
`[-rgb]`	RGB color index list
`[-color]`	Valid values are red green blue magenta yellow cyan and orange
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`[<objects>]`	Objects to unmark

Categories

[GUIControl](#)

Description

Unmarks the specified object or objects that were previously marked by the `mark_objects` command. This command is for use in GUI mode.

This command supports the color options as specified below. However, these options are not necessary to unmark a specific object, but can be used to unmark all objects currently marked in the specified color. See the example below.

Arguments

`-rgb` **<args>** - (Optional) The color to unmark in the form of an RGB code specified as {R G B}. For instance, {255 255 0} specifies the color yellow.

`-color <arg>` - (Optional) The color to unmark. Supported highlight colors are red, green, blue, magenta, yellow, cyan, and orange.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

Send Feedback

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<objects>` - (Optional) One or more objects to be unmarked. If no objects are specified, all marked objects of the specified color will be unmarked. If no color is specified, all marked objects will be unmarked.

Examples

The following example unmarks the selected objects:

```
unmark_objects [get_selected_objects]
```

The following example unmarks all objects currently marked in the color yellow:

```
unmark_objects -color yellow
```

See Also

- get_selected_objects
- mark_objects

Send Feedback

unplace_cell

Unplace one or more instances.

Syntax

```
unplace_cell [-quiet] [-verbose] <cell_list>...
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<cell_list>	a list of cells to be unplaced

Categories

[Floorplan](#)

Description

Unplace the specified cells from their current placement site.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the set_msg_config command.

<cell_list> - (Required) Specifies a list of one or more cells to be unplaced from the device.

Examples

The following example unplaces the specified cell:

```
unplace_cell {fftEngine/fftInst/ingressLoop[6].ingressFifo/buffer_fifo/
i_4773_12897}
```

The following example unplaces multiple cells:

```
unplace_cell {div_cntr_reg_inferredi_4810_15889  div_cntr_reg[0]
div_cntr_reg[1]}
```

See Also

- create_cell
- place_cell
- remove_cell

unregister_proc

Unregister a previously registered Tcl proc.

Syntax

```
unregister_proc [-quiet] [-verbose] <tasknm>
```

Returns

Nothing

Usage

Name	Description
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<tasknm>`	Name of Tcl task to unregister. The task must be wrapping a proc.

Categories

[Tools](#)

Description

Unregister the Tcl command, or <tasknm>, from the Vivado Design Suite Tcl interpretor.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<tasknm> - (Required) Name of Tcl task to unregister. The task must be wrapping a registered Tcl procedure.

Example

The following example unregisters the findCmd Tcl task:

```
unregister_proc findCmd
```

See Also

- register_proc

unselect_objects

Unselect items that are currently selected

Syntax

```
unselect_objects [-quiet] [-verbose] [<objects>]
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<objects>]	Objects to unselect

Categories

[GUIControl](#)

Description

Unselects the specified object or objects that were previously selected by the `select_objects` command.

This command will unselect both primary and secondary selected objects. The selection of secondary objects is controlled through the use of Selection Rules defined in the **Tools → Settings** command. Refer to the *Vivado Design Suite User Guide: Using the IDE (UG893)* for more information on Setting Selection Rules.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<objects>` - (Optional) One or more objects to be unselected. If no objects are specified, all selected objects will be unselected.

Send Feedback

Examples

The following example unselects the specified site on the device:

```
unselect_objects [get_sites SLICE_X56Y214]
```

The following example unselects all currently selected objects:

```
unselect_objects
```

See Also

- get_selected_objects
- select_objects

Send Feedback

update_bd_boundaries

update the boundary of source designs of a specified block design container cell from its active variant source design or a specified design. Otherwise, update the boundary of a block design from another block design provided

Syntax

```
update_bd_boundaries [-from_bd <arg>] [-check_only] [-quiet] [-verbose]
    [<of_objects>...]
```

Returns

TCL_OK on success, TCL_ERROR on failure.

Usage

Name	Description
`[-from_bd]`	block design to copy the boundary from
`[-check_only]`	report only the changes made when the boundary is copied, do not save the changes to disk
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`[<of_objects>]`	One or a list of block design container cells enabled for Dynamic Function Exchange or block desing names. List must be of one object type.

Categories

[IPIntegrator](#)

update_clock_routing

Update clock routing on global clocks if they are modified after placement

Syntax

```
update_clock_routing [-quiet] [-verbose]
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Tools

Description

The `update_clock_routing` command is an advanced command used for manually updating the routing structures of all global clocks in designs targeting the UltraScale architecture. The command operates on all global clocks in a design, not individual clocks.

Due to a more flexible clocking architecture, UltraScale and UltraScale+ designs require a two-step process for routing global clocks. First the Vivado placer assigns the routing resources required to route the global clocks from the clock source to the destination clock regions (CLOCK_ROOT or USER_CLOCK_ROOT). Next the Vivado router fills in the routing gaps on the clock nets. In between these two steps the resulting structures are called gap trees: each global clock net has its base routing resources assigned but with large routing gaps where no routing resources have been assigned.

After gap trees are constructed, the router optimally routes the remaining clock network to all leaf-level primitives to fill in the routing gaps. During an implementation run the global clock routing is handled automatically. However in cases where the clock tree has been changed after implementation, by modifying the USER_CLOCK_ROOT property on a clock net for instance, the Vivado tool may need the `update_clock_routing` command to properly rebuild the gap trees and fill in the routing gaps.

Examples of this include:

- Moving the clock root of a global clock.

- Adding or moving loads of a global clock into a clock region not yet occupied by the global clock, then running timing analysis on the updated design.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example updates the USER_CLOCK_ROOT property on the specified clock nets, unroutes the nets, and then updates the clock routing:

```
set_property USER_CLOCK_ROOT X1Y0 [get_nets {clk1 clk2}]
route_design -unroute -nets [get_nets {clk1 clk2}]
update_clock_routing
```

IMPORTANT! *The unroute command is needed to clean out existing clock routing on the clock nets before updating the clock routing.*

See Also

- route_design
- set_property

update_compile_order

Updates a fileset compile order and possibly top based on a design graph.

Syntax

```
update_compile_order [-force_gui] [-fileset <arg>] [-quiet] [-verbose]
```

Usage

Name	Description
[-force_gui]	Execute this command, even when run interactively in the GUI.
[-fileset]	Fileset to update based on a design graph
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Project

Description

Update the compile order of the design sources in the current project, or in the specified fileset.

Arguments

-force_gui - Update the compile order in GUI mode of the Vivado IDE.

-fileset <arg> - Update the compile order of the specified fileset.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the set_msg_config command.

Examples

The following example updates the compile order of the source files in the simulation fileset:

```
update_compile_order -fileset sim_1
```

See Also

- add_files
- import_files

Send Feedback

update_design

update the netlist of the current design

Syntax

```
update_design -cells <args> [-strict] [-from_file <arg>]
    [-from_design <arg>] [-from_cell <arg>] [-black_box] [-buffer_ports]
    [-quiet] [-verbose]
```

Usage

Name	Description
-cells	List of cells to update with a new sub-netlist.
[-strict]	Require exact ports match for replacing cell (otherwise extra ports are allowed).
[-from_file]	Name of the file containing the new sub-netlist.
[-from_design]	Name of the an open netlist design containing the new sub-netlist.
[-from_cell]	Name of cell in the from_design which defines the new sub-netlist.
[-black_box]	Update the cell to a black_box.
[-buffer_ports]	buffer all the ports of black box
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Project

Description

This command updates the in-memory design, replacing the current netlist in the specified cell with a netlist from a specified file, from another open design, from a specified cell of a design, or converts the cell to a black box cell.

The `update_design` command can update a single instance, or can update all instances of a master cell.

Only the in-memory view of the design is changed by the new netlist. You must save the design using the `write_checkpoint` command, or any updates will be lost when you close the project or exit the tool.

Send Feedback

Arguments

`-cells` <args> - (Required) Defines a list of names, or cell objects, to update with the specified netlist.

`-strict` - (Optional) Require the new netlist to have exactly the same ports as the cell it is imported into. The tool will perform some checking on the new netlist to insure that the specified netlist has all the ports required for the specified cell. However, additional ports are also permitted, unless the `-strict` option is used.

`-from_file` - (Optional) Name of a file containing the new netlist. The netlist can be in the form of a structured Verilog netlist (.v) or an EDIF netlist (.edf) file.

Note: `-from_file` and `-from_design` are mutually exclusive.

`-from_design` - (Optional) Allows you to import the netlist from another open design in the current project. The design must be opened in the current tool invocation, and not a separate process.

`-from_cell` - (Optional) Name of a cell in the design specified with `-from_design`. The netlist from the specified cell will be used to update the cell in the current design. By default the tool will use the top-level cell of the design specified in `-from_design`.

Note: This option can only be used with `-from_design`.

`-black_box` - (Optional) Change the specified cell into a black box cell.

`-buffer_ports` - (Optional) Insert buffers for all the ports of a black box cell.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

This example replaces a black box cell with the netlist from the specified file:

```
update_design -from_file C:/Data/cell_contents.v -cell black_box_cell
```

The following example updates the netlist in the arnd4 cell with the specified Verilog netlist:

```
update_design -cell arnd4 -from_file C:/Data/round_4.v
```

Send Feedback

The following example updates the arnd4 cell in the current design with the netlist from the same cell in the specified design:

```
update_design -cell arnd4 -from_design netlist_2 -from_cell arnd4
```

See Also

- get_cells
- write_checkpoint

update_files

Update file(s) in the project based on the file(s) or directory(ies) specified

Syntax

```
update_files [-from_files <args>] [-norecurse] [-to_files <args>]
    [-filesets <args>] [-force] [-report_only] [-quiet] [-verbose]
```

Returns

List of the files updated.

Usage

Name	Description
[-from_files]	New files and directories to use for updating
[-norecurse]	Recursively search in specified directories
[-to_files]	Existing project files and directories to limit updates to
[-filesets]	Fileset name
[-force]	Overwrite imported files in the project, even if read-only, if possible
[-report_only]	Do no actual file updates, but report on updates that otherwise would have been made
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Project

Description

Updates the specified files with the contents of specified remote files. Use this command to update a local file with the contents of its original remote file, or replace it with the contents of a different remote file.

This command returns a list of updated files, or returns an error if it fails.

Arguments

`-from_files` <args> - (Optional) An ordered list of files or directories to use when updating the `-to_files` to be updated.

`-norecurse` - (Optional) Disable recursive searching through specified sub-directories.

Send Feedback

-to_files <args> - (Optional) The path and filename of the file or files to update with the specified -from_files.

-filesets <args> - (Optional) Overwrite the files in the specified fileset with the -from_files.

-force - (Optional) Force the overwrite of specified files, even if they are write restricted.

-report_only - (Optional) Run the command a generate a report related to updated files, but do no actually update the files.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the set_msg_config command.

Examples

The following example updates the various project source IP core files with the specified -from_files, reporting the results without making any updates:

```
update_files -from_files C:/Data/IP/*.xci \
    -to_file [get_files *.xci} -report_only
```

Note: No warnings will be issued for newer local files that will be overwritten.

See Also

- reimport_files

update_hw_firmware

Update the SmartLynq firmware image

Syntax

```
update_hw_firmware [-file_path <arg>] [-config_path <arg>] [-skip_update]
    [-reset] [-format] [-flash] [-quiet] [-verbose] [<hw_server>]
```

Usage

Name	Description
[-file_path]	Optional path to BOOT.BIN file Default: Use default BOOT.BIN
[-config_path]	Optional path to config.ini file Default: No config.ini updated
[-skip_update]	Skip writing the BOOT.BIN to the SmartLynq
[-reset]	Reset the SmartLynq cable after any other operations to complete the update and disconnects the hw_server
[-format]	Format the SmartLynq cable EMMC prior to any other operations. Any files on SmartLynq cable will be lost.
[-flash]	Flash the SmartLynq cable QSPI. This updates the primary FSBL and Safe Mode images.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<hw_server>]	hardware server Default: current hardware server

Categories

[Hardware](#)

update_hw_gpio

Update the SmartLynq GPIO PMOD pin values

Syntax

```
update_hw_gpio [-quiet] [-verbose] [<output_enable_mask>]
     [<output_pin_values>] [<hw_server>]
```

Returns

All GPIO PMOD pin values.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<output_enable_mask>]	8 bit hex mask to enable output pins on GPIO Default: All output pins disabled
[<output_pin_values>]	8 bit hex value of output pins Default: All output pins driven low
[<hw_server>]	hardware server Default: current hardware server

Categories

Hardware

update_ip_catalog

Update the IP Catalog. Before executing this command optionally use the following to set repository paths:'set_property ip_repo_paths <repo_path_list> [current_fileset]'

Syntax

```
update_ip_catalog [-rebuild] [-add_ip <arg>] [-delete_ip <arg>]
    [-delete_mult_ip <args>] [-disable_ip <arg>] [-enable_ip <arg>]
    [-add_interface <arg>] [-create_index] [-repo_path <arg>]
    [-update_module_ref] [-quiet] [-verbose]
```

Returns

True for success.

Usage

Name	Description
[-rebuild]	Trigger a rebuild of the specified repository's index file or rebuild all repositories if none specified
[-add_ip]	Add the specified IP into the specified repository Values: Either a path to the IP's component.xml or to a zip file containing the IP
[-delete_ip]	Remove the specified IP from the specified repository Values: Either a path to the IP's component.xml or its VLNV
[-delete_mult_ip]	Remove the specified IPs from the specified repository Values: A list of IPs; either paths to the component.xml files or their VLNVs
[-disable_ip]	Disable the specified IP from the specified repository Values: Either a path to the IP's component.xml or its VLNV
[-enable_ip]	Enable the specified disabled IP from the specified repository Values: Either a path to the IP's component.xml or its VLNV
[-add_interface]	Add the specified interface into the specified repository Values: A path to the interface's xml file
[-create_index]	Cache the specified repository's data on disk, to improve load time.
[-repo_path]	Used in conjunction with rebuild, add_ip, delete_ip, delete_mult_ip, disable_ip or create_index to specify the path of the repository on which to operate
[-update_module_ref]	Update module reference from their source (e.g. HDL file)
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

IPFlow

Description

Update the IP catalog associated with the current design.

The Xilinx® IP catalog, or repository, is located in the installation hierarchy of the Vivado Design Suite software release being used. You can also add custom IP to the repository by using the `set_property` command to set the IP_REPO_PATHS property on the source fileset to point to the locations of custom IP, as shown in the example below.

The `update_ip_catalog` command lets you add, delete, disable, or enable individual IP cores in the catalog. When referring to individual cores, you can reference them by the path to the `component.xml` file, or by referencing the VLNV property of the IP.

> 💡 **TIP:** *The VLNV property refers to the Vendor:Library:Name:Version string which uniquely identifies the IP in the catalog.*

This command returns a transcript of its process if successful, or returns an error if it fails.

Arguments

`-rebuild` - (Optional) Rebuild the complete IP catalog index, or just rebuild the index for the IP repository specified by the `-repo_path`.

`-add_ip <arg>` - (Optional) Add an individual IP core to the specified IP repository. This argument requires the `-repo_path` argument to also be specified. The IP is specified as a path to the `component.xml` of the IP, or the path to a zip file containing the IP.

`-delete_ip <arg>` - (Optional) Remove an IP core from the specified IP repository. This argument requires the `-repo_path` argument to also be specified. The IP is specified as a path to the `component.xml` of the IP, or as the VLNV property of the IP.

`-delete_mult_ip <arg>` - (Optional) Remove the specified IP cores from the IP repository. This argument requires the `-repo_path` argument to also be specified. The IPs are specified either as paths to the `component.xml` files, or as the VLNV properties of the IP.

`-disable_ip <arg>` - (Optional) Disable an IP core from the specified IP repository. This argument requires the `-repo_path` argument to also be specified. The IP is specified as a path to the `component.xml` of the IP, or as the VLNV property of the IP.

`-enable_ip <arg>` - (Optional) Enable a previously disabled IP core from the specified repository. This argument requires the `-repo_path` argument to also be specified. The IP is specified as a path to the `component.xml` of the IP, or as the VLNV property of the IP.

`-add_interface <arg>` - (Optional) Specify the path to the XML file of a user-defined AXI interface to add to the IP repository.

`-create_index` - (Optional) Cache the specified repository's data on disk, to improve load time. This argument requires the `-repo_path` argument to also be specified.

`-repo_path <arg>` - (Optional) Used in conjunction with `-rebuild`, `-add_ip`, `-delete_ip`, `-delete_mult_ip` or `-create_index` to specify the directory name of an IP repository to operate on.

> **IMPORTANT!** *The IP repository must have been previously added to the current source fileset using the* `set_property` *command to set the IP_REPO_PATH. See the example below.*

`-update_module_ref` - (Optional) Refresh the block design cell or cells that reference module definitions from RTL source files by rereading a module definition from the source file.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example sets the IP_REPO_PATHS property of the current Source fileset, to add an IP repository, then rebuilds the IP catalog index for the whole IP catalog:

```
set_property IP_REPO_PATHS C:/Data/IP_LIB [current_fileset]
update_ip_catalog -rebuild
```

This example disables the IP specified by its VLNV property from the specified IP repository:

```
update_ip_catalog -disable_ip {myCo.com:ip:custom_decoder:1.0} \
    -repo_path C:/Data/ip
```

This example disables the IP specified by the path to the `component.xml` file, from the IP repository:

```
update_ip_catalog -disable_ip C:/Data/ip/custom_encoder_1/component.xml \
-repo_path C:/Xilinx/Vivado/data/ip
```

See Also

- create_ip
- import_ip
- generate_target

Send Feedback

- update_module_reference
- validate_ip

update_macro

Update a macro

Syntax

```
update_macro [-absolute_grid] [-quiet] [-verbose] <macro> <rlocs>
```

Usage

Name	Description
[-absolute_grid]	Use absolute grid for relative locations
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<macro>	Macro to update
<rlocs>	a list interleaved instances and site names

Categories

XDC

Description

Populate a previously created macro with leaf cells and relative placements.

A macro is made up of primitive, or leaf-level logic cells, and their associated connections, positioned in a placement grid. The specified relative locations, or <rlocs>, can be based on a relative grid, or on an absolute grid, called an RPM_GRID. Refer to the *Vivado Design Suite User Guide: Implementation (UG904)* for more information on absolute and relative placement grids

A cell can only belong to a one macro. If you attempt to assign a leaf-level cell to multiple macros, the Vivado tool will return an error. If you attempt to assign a non-primitive cell to a macro, the tool will return an error.

To change the contents of an existing macro, you must delete the macro with `delete_macro`, recreate it with `create_macro`, and update it with new contents. You cannot simply overwrite or modify an existing macro.

Arguments

`-absolute_grid` - (Optional) Use to indicate that the <rlocs> are specified in the absolute RPM_GRID.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Send Feedback

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<macro> - (Required) Specify the name of the macro to update.

<rlocs> - (Required) Specify the leaf-cells to include in the macro, and their relative locations (RLOCs). When `-absolute_grid` is specified, the location is based on actual device coordinates instead of relative locations. The cells and RLOCs are specified as name-value pairs, with multiple leaf-cells and RLOCs assigned to a single macro as follows:

```
{cell0 XmYn cell1 XmYn ... cellN XmYn}
```

Where:

- m = An integer representing the relative or absolute X coordinate of the preceding cell.

- n = An integer representing the relative or absolute Y coordinate of the preceding cell.

- Cell coordinates are relative to each other, unless the `-absolute_grid` option has been specified.

- The relative coordinates are based on a theoretical array of Slices, located relative to each other.

- The absolute coordinates are determined by the RPM_X and RPM_Y properties from actual Slices of the target device. These can be determined by the `report_property` command for selected sites.

Examples

The following example creates a macro named usbMacro0, sets the current instance to the usbEngine0/u0 module, assigns three cells to the macro, with a relative placement for each cell to have two of them placed inside the same Slice, and the third placed in a vertically adjacent Slice:

```
create_macro usbMacro0
current_instance usbEngine0/u0
update_macro usbMacro0 {rx_active_reg X0Y0 \
    rx_err_reg X0Y0 rx_valid_reg X0Y1}
```

Send Feedback

The following example creates a macro named usbMacro1, assigns three cells to the macro using the hierarchical path to the leaf-cell, with absolute coordinates specified for the cells in the macro:

```
create_macro usbMacro1
set Site1 [get_sites SLICE_X8Y77]
set Site2 [get_sites SLICE_X9Y77]
set Site3 [get_sites SLICE_X8Y78]
set RPM1 X[get_property RPM_X $Site1]Y[get_property RPM_Y $Site1]
set RPM2 X[get_property RPM_X $Site2]Y[get_property RPM_Y $Site2]
set RPM3 X[get_property RPM_X $Site3]Y[get_property RPM_Y $Site3]
update_macro usbMacro1 -absolute_grid "usbEngine1/u0/rx_active_reg $RPM1 \
usbEngine1/u0/rx_err_reg $RPM2 usbEngine1/u0/rx_valid_reg $RPM3 "
```

Note: In the prior example, notice the use of Tcl variables to capture the Sites of interest, and extract the RPM_X and RPM_Y properties of those sites for use in the update_macro command. Also notice the use of quotes ("") instead of curly braces ({}) in the update_macro command. This is to allow the Tcl shell to perform variable substitution of the command. Refer to the *Vivado Design Suite User Guide: Using Tcl Scripting (UG894)* for more information on variables and variable substitution.

This command reports the properties on the usbMacro1 macro to see the absolute grid coordinates assigned to the cells in the macro:

```
report_property -all [get_macros usbMacro1]
```

See Also

- create_macro
- delete_macros
- get_macros
- get_property
- get_sites
- place_design
- report_property

update_module_reference

Refresh module reference definition and instance(s)

Syntax

```
update_module_reference [-quiet] [-verbose] [<ips>...]
```

Returns

A return code indicating success or failure.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<ips>]	module reference to be upgraded Values: IP instance name(s) within the design

Categories

IPFlow

Description

Refresh the block design cell or cells that reference module definitions from RTL source files by rereading the module definition from the source file.

Note: This command does not cause the Vivado tool to reread the source file. If changes have been made to the source file it must be separately updated.

This command returns a transcript of the update process as well as any warnings related to design changes, or returns nothing if the command fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Send Feedback

Note: Message limits can be defined with the `set_msg_config` command.

<ips> - (Optional) Specifies the module references in the current design to upgrade. Modules can be specified by name, or returned by the `get_ips` command.

Examples

The following example updates the specified module references in the current design:

```
update_module_reference {rtlRam_0 uart_0}
```

See Also

- create_bd_cell
- create_bd_design

update_noc_qos

Update the NoC solution

Syntax

```
update_noc_qos [-force] [-quiet] [-verbose]
```

Usage

Name	Description
[-force]	Force update even if the existing solution is invalid
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Tools

update_sw_parameters

Update the dcp by adding the generated sw param info files to it

Syntax

```
update_sw_parameters [-quiet] [-verbose]
```

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Project

Description

Updates the design check-points (DCPs) with the latest hardware def. This hardware def will have the updated sw parameters information.

Whenever a software parameter of an IP is modified, the runs need not to go stale. But already generated design checkpoints need to be updated with the modified parameter information. This information is captured in the hwdef file. With update_sw_parameters command, the hwdef file is added to already generated design checkpoints.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the set_msg_config command.

Examples

The following example modifies a software parameter for the CIPS IP, generates the hw_handoff target and then updates the DCPs with the modified values:

```
set_property -dict [list CONFIG.PS_I2C0_PERIPHERAL_ENABLE {1}]
[get_bd_cells versal_cips_0]
generate_target hw_handoff [get_files top.bd]
update_sw_parameters
```

See Also

- refresh_design
- close_design

Send Feedback

update_timing

update timing

Syntax

```
update_timing [-full] [-quiet] [-verbose]
```

Usage

Name	Description
[-full]	Perform a full timing update instead of an incremental one
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Timing

Description

Updates timing for the current design.

Update the timing data to reflect any timing constraints that were added to the design since the timing engine was last run. This command updates the in-memory view of the timing database, without incurring the time of a full timing analysis.

Timing is automatically updated by commands that change timing or need updated timing information, such as the report_timing command. The update_timing command lets you manually trigger the timing update to insure the latest constraints are applied to the timing engine.

The update_timing command uses an incremental analysis approach by default, which updates only out-of-date information, to reduce process and analysis time. You can also specify a complete or full update to insure a comprehensive review of timing data in the design. However, to avoid long timing analysis run times, you should use the -full option only when you need to.

Arguments

-full - (Optional) Perform a complete timing analysis rather than the default incremental update of the timing data.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Send Feedback

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example performs a full update of the in-memory timing data:

```
update_timing -full
```

See Also

- report_timing
- report_timing_summary
- reset_timing

Send Feedback

upgrade_bd_cells

Upgrade configurable IPIntegrator cell(s) to later version

Syntax

```
upgrade_bd_cells [-latest <arg>] [-quiet] [-verbose] <objects>...
```

Returns

List of IPIntegrator cell names those were upgraded, "" if failed.

Usage

Name	Description
[-latest]	Upgrade the IPIntegrator block to the latest version
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<objects>	IPIntegrator cells to be upgraded

Categories

IPIntegrator

Description

Upgrade IP integrator cells to the latest version available in the IP integrator catalog.

This command lets you update IP integrator subsystem designs from an earlier release to use the IP cores from the latest catalog.

This command returns the list of IP integrator cells that were upgraded, or returns an error if it fails.

Arguments

-latest - (Optional) Upgrade the specified cells to the latest available version from the IP catalog. This option is the default, and is not required.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

Send Feedback

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the set_msg_config command.

<objects> - The IP integrator cells to upgrade to the latest version. These objects must be specified using the get_bd_cells command. Objects cannot be referenced by name.

Example

The following example upgrades the specified cells to the latest version available in the IP integrator catalog:

```
upgrade_bd_cells [get_bd_cells {vidOut1 cmpy_1 newMod1}]
INFO: [BD 41-1162] The cell '/vidOut1' is already at its latest version.
INFO: [BD 41-1162] The cell '/cmpy_1' is already at its latest version.
WARNING: [BD 41-1082] Hierarchy block (/newMod1) cannot be directly
upgraded.
Please dive into the hierarchy and select individual cells to upgrade.
```

Note: A warning message is returned for the user-defined hierarchical module.

See Also

- [create_bd_cell](#)
- [get_bd_cells](#)

Send Feedback

upgrade_ip

Upgrade a configurable IP to a later version

Syntax

```
upgrade_ip [-srcset <arg>] [-vlnv <arg>] [-log <arg>] [-quiet] [-verbose]
    <objects>...
```

Returns

A return code indicating success or failure.

Usage

Name	Description
[-srcset]	(Optional) Specifies the source file set containing the IP to be upgraded Default: The current source fileset Values: Source set name
[-vlnv]	(Optional) Identifies the Catalog IP to which the IP will be upgraded. The VLNV string maps to the IPDEF property on the IP core. This is a strict comparison, and the upgrade will fail if the identified IP does not exist in the Catalog. Default: Latest version of the current IP Values: A string of the form '<vendor>:<library>:<name>:<versio n>'
[-log]	(Optional) Identifies the log file to which the IP upgrade report will be concatenated. Default: An empty string, indicating that no log will be written Values: A file path to an existing writable file, or a non-existent file location in a writable directory
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<objects>	IP to be upgraded Values: IP instance(s) within the design, as returned by 'get_ips <instance name>' or 'get_bd_cells <cell name>'

Categories

[IPFlow](#)

Description

This command upgrades the specified IP cores from an older version to the latest version in the IP catalog.

You can only upgrade IP that explicitly supports upgrading. The UPGRADE_VERSIONS property on the ipdef object indicates if there are upgrade versions for an IP core.

> 💡 **TIP:** *The `upgrade_ip` command also accepts Block Design cell IP instances as bd_cell objects. The command upgrades the bd_cell objects within the Block Design, and does not require the diagram to be open in the Vivado IP integrator .*

Arguments

`-srcset <arg>` - (Optional) Specifies the source file set to upgrade the IP files in. If not specified, the default source file set is `sources_1`.

`-vlnv <arg>` - (Optional) Specify the Vendor:Library:Name:Version attribute of the IP to upgrade from the IP catalog. The VLNV attribute identifies the object in the IP catalog.

`-log <arg>` - (Optional) Specifies the name of a file to append the IP upgrade information to. By default the upgrade_ip command does not log its activities.

Note: If the path is not specified as part of the file name, the log file will be written into the current project directory.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<objects>` - (Required) Specifies which IP cores of Block Design cells to upgrade. The IP must be specified as objects returned by the `get_ips` command.

> ⭐ **IMPORTANT!** *Do not use the `get_ips -all` option, as this can result in recursion issues.*

Examples

The following example upgrades all IP cores in the current project to the latest version:

```
upgrade_ip [get_ips]
```

See Also

- create_ip
- get_bd_cells
- get_ips
- import_ip
- open_bd_design

Send Feedback

upgrade_project

Upgrade a project from a previous release to use the latest Vivado features.

Syntax

```
upgrade_project [-migrate_output_products] [-log <arg>] [-quiet] [-verbose]
```

Usage

Name	Description
[-migrate_output_products]	Migrate the output products directory structure to the latest Vivado structure.
[-log]	(Optional) Identifies the log file to which the project upgrade report will be concatenated. Default: An empty string, indicating that no log will be written Values: A file path to an existing writable file, or a non-existent file location in a writable directory
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Project](#)

upload_hw_ila_data

Stop capturing. Upload any captured hardware ILA data.

Syntax

```
upload_hw_ila_data [-quiet] [-verbose] [<hw_ilas>...]
```

Returns

Hardware ILA data objects.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<hw_ilas>]	List of hardware ILA objects. Default: Current hardware ILA

Categories

[Hardware](#)

Description

Upload the captured data from the memory buffers of the specified ILA debug cores on the Xilinx FPGA hardware device, and move it into a hw_ila_data object in the Vivado logic analyzer.

You can upload captured data from the ILA debug core at any time during the capture process triggered by the `run_hw_ila` command. However, you may want to use the `wait_on_hw_ila` command in any Tcl scripts, to wait until the sample data buffers of the ILA core are fully populated with data. If you run the `upload_hw_ila_data` command prior to this, you may see a message as follows:

```
INFO: [Labtools 27-1965] The ILA core 'hw_ila_1' trigger was stopped by
user \
at 2014-Mar-06 08:59:30
INFO: [Labtools 27-2212] The ILA core 'hw_ila_1' captured '6' windows with \
'64' samples each, and a last partial window with '0' samples.
```

The upload process creates a hw_ila_data object in the process of moving the captured data from the ILA debug core, hw_ila, on the physical FPGA device, hw_device. The hw_ila_data object is named after the hw_ila core it is uploaded from.

> 💡 **TIP:** *Each hw_ila object has only one matching hw_ila_data object associated with it. Each time upload_hw_ila_data is run for a specific hw_ila core, the hw_ila_data object is overwritten if it already exists.*

The data object, hw_ila_data can be viewed in the waveform viewer of the Vivado logic analyzer by using the `display_hw_ila_data` command, and can be written to disk using the `write_hw_ila_data` command.

This command returns a hw_ila_data object, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_ilas> - (Optional) Specify one or more hw_ila objects to upload data from. The hw_ila objects can either be specified as objects returned by the `get_hw_ilas` or `current_hw_ila` commands, or specified by name. If the hw_ila is not specified, the data will be uploaded from the `current_hw_ila`.

Example

The following example arms the current hardware ILA debug core on the target hw_device, captures sample data at the probes as trigger events or capture conditions are encountered. Tcl script processing is suspended while sample data is captured, and then the data is uploaded from the hw_ila on the hw_device, into a hw_ila_data object:

```
run_hw_ila -trigger_now [current_hw_ila]
wait_on_hw_ila [current_hw_ila]
upload_hw_ila_data [current_hw_ila]
```

See Also

- current_hw_device
- current_hw_ila
- current_hw_ila_data
- display_hw_ila_data
- get_hw_devices
- get_hw_ilas

Send Feedback

- get_hw_ila_datas
- run_hw_ila
- wait_on_hw_ila
- write_hw_ila_data

validate_bd_design

Run Parameter Propagation for specified design or for a specific cell.

Syntax

```
validate_bd_design [-force] [-design <arg>] [-include_pfm] [-steps <args>]
    [-quiet] [-verbose]
```

Returns

TCL_OK, TCL_ERROR if failed.

Usage

Name	Description
[-force]	Force re-run validation on the design
[-design]	Design name. If not specified, run parameter propagation on current design
[-include_pfm]	including validate pfm attributes on the design
[-steps]	List of steps to execute. This does not put the design in a validated state. Choices: addressing,noc-drc,noc-compile
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

IPIntegrator

Description

Validate an IP integrator subsystem design, or IP cell or hierarchical module.

Arguments

-force - (Optional) Force validation on the block design.

-design <arg> - (Optional) The IP integrator subsystem design to validate. If not specified, the current IP integrator subsystem design is validated.

-include_pfm - (Optional) Validate the required DSA and PFM attributes on a block design created as the hardware portion of a Vitis platform. Errors will be returned if the platform properties are not properly defined.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Send Feedback

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example validates the current IP integrator subsystem design, forcing re-validation if needed:

```
validate_bd_design -force
```

See Also

- create_bd_design
- open_bd_design
- save_bd_design

validate_board_files

Check whether the XML files describing a board in the given directory are valid. Only supported for board XML files with schema_version>=2.0. The XML files must contain an appropriate DOCTYPE declaration to be fully validated. Examples: <!DOCTYPE board SYSTEM "board.dtd"> <!-- for board.xml --> <!DOCTYPE ip_presets SYSTEM "preset.dtd"> <!-- for preset.xml --> <!DOCTYPE part_info SYSTEM "part0_pins.dtd"> <!-- for part0_pins.xml -->

Note that if a project is open and an IP repository loaded, this command will also validate certain IP and Interface attributes used in the board definition against the current IP repository.

Syntax

```
validate_board_files [-quiet] [-verbose] [<dir>...]
```

Returns

Ok if all board files are valid.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<dir>]	The name of a directory containing the board files (board.xml, part0_pins.xml, preset.xml) to be checked

Categories

Object, Project, Board

Send Feedback

validate_cluster_configurations

Validate list of cluster configurations

Syntax

```
validate_cluster_configurations [-quiet] [-verbose]
    <cluster_configurations>
```

Usage

Name	Description
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<cluster_configurations>`	List of cluster configurations

Categories

cluster_configuration

Description

Validates the list of cluster configurations provided as an input. Use the `get_cluster_configurations` command to get a list of cluster configurations.

`validate_cluster_configurations` validates the submit and kill command of the cluster configuration objects to ensure if a job can be submitted using a particular cluster configuration or not.

Currently, Vivado supports following clusters management tools.

- Load Sharing Facility (LSF)

- Sun Grid Engine (SGE)

- Simple Linux Utility For Resource Management (SLURM)

Arguments

`<cluster_configurations>` - (Required) List of cluster configurations object to be removed. The `get_cluster_configurations` command can be used to get the list of cluster configurations and provide it as input to the `remove_cluster_configurations` command.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example validates all the user defined cluster configurations.

```
validate_cluster_configurations [get_cluster_configurations -filter
{IS_DEFAULT == 0}]
```

The following example validates a cluster configuration named *lsf_medium*.

```
validate_cluster_configurations [get_cluster_configurations lsf_medium]
```

See Also

- create_cluster_configuration
- get_cluster_configurations
- remove_cluster_configurations

validate_hw_platform

Validate the specified harware platform

Syntax

```
validate_hw_platform [-verbose] [-quiet] [<file>]
```

Returns

The name of the shell file.

Usage

Name	Description
[-verbose]	Dump verbose information
[-quiet]	Ignore command errors
[<file>]	Xilinx Shell Archive file Values: Path to shell file.

Categories

Vitis, Platform

Description

Validate a Xilinx support archive (XSA) file to ensure it has the required content for the platform.

This command returns information related to the validation of the XSA, or returns an error if it fails.

Arguments

`-verbose` - (Optional) Provide detailed information related to the XSA and the validation process.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`<file>` - (Optional) Specify the path and file name of the XSA file.

Note: If the path is not specified as part of the file name, the tool will search for the specified file in the current working directory and then in the directory from which the tool was launched.

Send Feedback

Examples

The following example validates the specified XSA, returning verbose information related to the platform contents:

```
validate_hw_platform C:/Data/zc706.xsa -verbose
```

See Also

- open_hw_platform
- write_hw_platform

Send Feedback

validate_ip

This command applies any pending set_property commands and returns parameter validation messages, if any exist

Syntax

```
validate_ip [-save_ip] [-quiet] [-verbose] [<ips>]
```

Usage

Name	Description
`[-save_ip]`	Write IP files on the disk
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`[<ips>]`	IPs to be validated

Categories

IPFlow

Description

Perform DRC check on IP to ensure that it is properly constructed. This command returns 1 when all IPs have been validated, or 0 when there is a problem.

Arguments

`-save_ip` - (Optional) Updates the existing IP files after validation. No new files are created but the XCI and BOM files for the core are updated.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<ips> - (Optional) Specifies the set of IP cores to be validated.

Send Feedback

Examples

The following example validates the IPs in the current project, and updates the persistent representation of the IP.

```
validate_ip -save_ip [get_ips]
```

See Also

- create_ip
- generate_target
- upgrade_ip
- update_ip_catalog
- import_ip

verify_hw_devices

Verify hardware devices

Syntax

```
verify_hw_devices [-key <arg>] [-user_efuse <arg>] [-control_efuse <arg>]
     [-security_efuse <arg>] [-verbose] [-quiet] [<hw_device>...]
```

Returns

Hardware devices

Usage

Name	Description
`[-key]`	option value for key verification: efuse
`[-user_efuse]`	hex user fuse value for verification
`[-control_efuse]`	hex control fuse value for verification
`[-security_efuse]`	hex security fuse value for verification
`[-verbose]`	Shows fuse values during verification
`[-quiet]`	Ignore command errors
`[<hw_device>]`	list of hardware devices Default: current hardware device

Categories

Hardware

Description

For EFUSE encrypted devices, this command compares the bitstream assigned to the PROGRAM.FILE property on the specified hw_device with the bitstream programmed into the device with the `program_hw_devices` command.

Filtered through a required mask file, associated with the hw_device, the `verify_hw_devices` command uses both the bitstream and mask file to compare only the bits that are marked as important in the mask file. A mask file can be created along with the bitstream using the `write_bitstream` command, and is associated with the hw_device using the `create_hw_bitstream` command.

> **IMPORTANT!** *Verification cannot be performed on devices programmed with encrypted bitstreams, other than to verify that the* `-key` *has been programmed.*

The `verify_hw_devices` command reports that the readback data matches the programmed bitstream if successful, or returns an error if it fails.

Arguments

`-key efuse` - (Optional) Verify the encryption key is programmed on the specified hw_device in eFUSE registers.

`-user_efuse <arg>` - (Optional) Verify the provided HEX value is programmed into the FUSE_USER register on the hw_device.

`-control_efuse <arg>` - (Optional) Verify the provided HEX value is programmed into the FUSE_CNTL register on the hw_device.

`-security_efuse <arg>` - (Optional) Verify the provided HEX value is programmed into the FUSE_SEC register on the hw_device.

`-verbose` - (Optional) Report eFUSE register values when verifying the device.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

<hw_device> - (Optional) Specify one or more hw_device objects to verify. The hw_device must be specified as an object as returned by the `get_hw_devices` command. If the device is not specified, the `current_hw_device` will be verified.

Example

The following example verifies the bitstream on current hardware device:

```
verify_hw_devices [current_hw_device]
```

See Also

- connect_hw_server
- create_hw_device
- create_hw_target
- current_hw_device
- current_hw_target
- get_hw_devices
- get_hw_targets
- open_hw_target
- program_hw_devices
- write_bitstream
- write_hw_svf

version

Returns the build for Vivado and the build date

Syntax

```
version [-short] [-quiet] [-verbose]
```

Returns

Vivado version

Usage

Name	Description
[-short]	Return only the numeric version number
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Report

Description

Returns the version number of the Xilinx® tool. This includes the software version number, build number and date, and copyright information.

Arguments

`-short` - (Optional) Returns the version number of the software only.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

Examples

The following example returns only the version number for the software:

```
version -short
```

Send Feedback

wait_on_hw_ila

Wait until all hardware ILA data has been captured.

Syntax

```
wait_on_hw_ila [-timeout <arg>] [-quiet] [-verbose] [<hw_ilas>...]
```

Usage

Name	Description
[-timeout]	Timeout in minutes. Decimal value allowed Default: No timeout
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<hw_ilas>]	hardware ILA objects. Default: Current hardware ILA

Categories

[Hardware](#)

Description

Suspend Tcl script or Tcl command processing until the ILA debug core memory is filled by captured data samples.

This command is used after the `run_hw_ila` command to pause Tcl processing to wait for the data buffers to fill up. When the `wait_on_hw_ila` command returns, the Tcl command or script processing can continue.

With the ILA debug core memory filed with sample data, when Tcl processing resumes, you can upload the captured data samples into an ILA debug core data object, or hw_ila_data object. Use the `upload_ila_data` command to perform this action.

This command operates silently, returning nothing if successful, or returning an error if it fails.

Arguments

`-timeout <arg>` - (Optional) Wait for this period of time for all data on the ILA debug cores to be captured. If the timeout interval is reached, the wait command is terminated.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Send Feedback

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_ilas> - (Optional) Specify one or more hw_ila objects to wait on. The hw_ila objects must be specified as an object as returned by the `get_hw_ilas` or `current_hw_ila` commands. If the hardware ILA is not specified, the `current_hw_ila` will be run.

Example

The following example waits for all data on the current hardware ILA debug core to be captured:

```
run_hw_ila hw_ila_1 -trigger_now 1
INFO: [Labtools 27-1964] The ILA core 'hw_ila_1' trigger was armed
   at 2014-Mar-02 13:20:30
wait_on_hw_ila hw_ila_1
display_hw_ila_data [upload_hw_ila_data hw_ila_1]
INFO: [Labtools 27-1966] The ILA core 'hw_ila_1' triggered
   at 2014-Mar-02 13:20:31
```

See Also

- current_hw_device
- current_hw_ila
- current_hw_ila_data
- display_hw_ila_data
- get_hw_devices
- get_hw_ilas
- run_hw_ila
- upload_hw_ila_data

Send Feedback

wait_on_hw_sio_scan

Wait until hardware SIO scan has completed.

Syntax

```
wait_on_hw_sio_scan [-timeout <arg>] [-quiet] [-verbose] <hw_sio_scans>...
```

Usage

Name	Description
[-timeout]	Timeout in minutes. Decimal value allowed Default: No timeout
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hw_sio_scans>	List of hardware SIO scan objects.

Categories

Hardware

Description

Suspend a Tcl script or Tcl command processing until the specified serial I/O analyzer scan is complete.

This command is used after the `run_hw_sio_scan` command to pause Tcl processing to wait for the scan to complete. When the `wait_on_sio_scan` command returns, the Tcl command or script processing can continue.

This command operates silently, returning nothing if successful, or returning an error if it fails.

Arguments

`-timeout <arg>` - (Optional) Wait for this period of time for the serial I/O analyzer scan to complete. If the timeout interval is reached, the wait command is terminated.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Send Feedback

Note: Message limits can be defined with the `set_msg_config` command.

<hw_sio_scans> - (Required) Specify one or more hw_sio_scan objects to wait on. The hw_sio_scan must be specified as an object as returned by the `create_hw_sio_scan` or `get_hw_sio_scans` commands.

Example

The following example waits for the serial I/O analyzer scan to complete:

```
run_hw_sio_scan [lindex [get_hw_sio_scans {SCAN_0}] 0]
wait_on_hw_sio_scan [lindex [get_hw_sio_scans {SCAN_0}] 0]
```

See Also

- create_hw_sio_scan
- current_hw_device
- get_hw_sio_scans
- remove_hw_sio_scan
- run_hw_sio_scan
- stop_hw_sio_scan
- write_hw_sio_scan

Send Feedback

wait_on_hw_sio_sweep

Wait until hardware SIO sweep has completed.

Syntax

```
wait_on_hw_sio_sweep [-timeout <arg>] [-quiet] [-verbose]
    <hw_sio_sweeps>...
```

Usage

Name	Description
[-timeout]	Timeout in minutes. Decimal value allowed Default: No timeout
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<hw_sio_sweeps>	List of hardware SIO sweep objects.

Categories

[Hardware](#)

Description

Suspend a Tcl script or Tcl command processing until the serial I/O analyzer sweep scan is complete.

This command is used after the `run_hw_sio_sweep` command to pause Tcl processing to wait for the sweep scan to complete. When the `wait_on_sio_sweep` command returns, the Tcl command or script processing can continue.

This command operates silently, returning nothing if successful, or returning an error if it fails.

Arguments

`-timeout <arg>` - (Optional) Wait for this period of time for the serial I/O analyzer sweep scan to complete. If the timeout interval is reached, the wait command is terminated.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<hw_sio_sweeps> - (Required) Specify one or more hw_sio_sweep objects to wait on. The hw_sio_sweep must be specified as an object as returned by the `create_hw_sio_sweep` or `get_hw_sio_sweeps` commands.

Example

The following example launches an SIO sweep scan and waits for the sweep to complete:

```
run_hw_sio_sweep [lindex [get_hw_sio_sweeps {SWEEP_0}] 0]
wait_on_hw_sio_sweep [lindex [get_hw_sio_sweeps {SWEEP_0}] 0]
```

See Also

- create_hw_sio_sweep
- current_hw_device
- get_hw_sio_sweeps
- remove_hw_sio_sweep
- run_hw_sio_sweep
- stop_hw_sio_sweep
- write_hw_sio_sweep

wait_on_run

Block execution of further Tcl commands until the specified run completes.

Syntax

```
wait_on_run [-timeout <arg>] [-quiet] [-verbose] <run>
```

Usage

Name	Description
[-timeout]	Maximum time to wait for the run to complete (in minutes) Default: -1
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<run>	Run to wait on

Categories

Project

Description

Blocks the execution of Tcl commands until the specified run has completed either successfully or in error, or until the specified amount of time has elapsed.

This command will tell you when the run has terminated, but not the results of the run. To determine if the run has completed successfully, you could query the value of the PROGRESS property of the run:

```
launch_runs synth_1
wait_on_run synth_1
if {[[get_property PROGRESS [get_runs synth_1]] != "100%"} {
    error "ERROR: synth_1 failed"
}
```

The wait_on_run command can be used for runs that have been launched. If the specified run has not been launched when the wait_on_run command is used, you will get an error. Runs that have already completed do not return an error.

Note: This command is used for running the tool in batch mode or from Tcl scripts. It is ignored when running interactively from the GUI.

Arguments

`-timeout <arg>` - (Optional) The time in minutes that the `wait_on_run` command should wait until the run finishes. This allows you to define a period of time beyond which the tool should resume executing Tcl commands even if the specified run has not finished execution. The default value of -1 is used if timeout is not specified, meaning that there is no limit to the amount of time the tool will wait for the run to complete.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<run>` - (Required) The name of the run to wait on.

Examples

The following example launches the impl_1 run, and then waits for the specified run to complete, or to wait for one hour, whichever occurs first:

```
launch_runs impl_1
wait_on_run -timeout 60 impl_1
```

See Also

- launch_runs

write_abstract_shell

Write an abstract shell checkpoint of the current design

Syntax

```
write_abstract_shell -cell <arg> [-force] [-quiet] [-verbose] <file>
```

Returns

The name of the checkpoint file.

Usage

Name	Description
-cell	Create an abstract shell for this reconfigurable cell
[-force]	Overwrite existing checkpoint file
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<file>	Design checkpoint file Values: A filename with alphanumeric characters and .dcp extension.

Categories

FileIO

write_bd_layout

Export layout in native, pdf or svg

Syntax

```
write_bd_layout [-force] [-format <arg>] [-orientation <arg>]
    [-scope <arg>] [-hierarchy <arg>] [-quiet] [-verbose] <file>
```

Returns

Name of the output file.

Usage

Name	Description
[-force]	Overwrite existing file
[-format]	Values: native, pdf or svg. regenerate_bd_layout -layout_file can be used with native layout. Default: native
[-orientation]	Values: landscape or portrait
[-scope]	Values: visible or all Default: all
[-hierarchy]	Hierarchy block
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<file>	Output file

Categories

[FileIO](#)

Description

Write the current open block design in the Vivado IP integrator to the specified file format.

This command lets you print the block design, output it as a vector graphic file for use in documentation related to the design project, or recreate the block design layout in the Vivado IP integrator design canvas using the `regenerate_bd_layout` command.

This command returns the name of the file written, or returns an error if it fails.

Arguments

`-force` - (Optional) Overwrite an existing file of the specified name.

`-format [ native | pdf | svg ]` - (Optional) Write the output file in the specified format. The default file format is `native`, and describes an internal Vivado tool format that can be used to recreate the block design layout in the Vivado IP integrator using the `regenerate_bd_layout` command. `SVG` is scalable vector graphics format. `PDF` is portable document format.

Note: `regenerate_bd_layout` can be used with the `native` format.

`-orientation [ landscape | portrait ]` - (Optional) Specify the orientation of the graphic file as either `landscape` (horizontal) or `portrait` (vertical) orientation. The default orientation is `portrait`.

`-scope [ visible | all ]` - (Optional) Defines the scope of the block design to output to the specified file. The default is `all`, and indicates to output the whole block design. The `visible` outputs only the currently displayed limits of the design canvas, or the current zoom value.

`-hierarchy <arg>` - (Optional) Output the drawing of the specified hierarchical bd_cell. Blocks can be specified by the `get_bd_cells` command.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file>` - (Required) Write the block design canvas into the specified file. If the specified file already exists, you must also use the `-force` option to overwrite it, or an error is returned.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

Examples

The following example prints the current block design to the specified PDF file:

```
write_bd_layout -format pdf -orientation landscape C:/Data/microblaze.pdf
```

The following example prints the specified hierarchical cell of the block design to the specified SVG file:

```
write_bd_layout -format svg -orientation landscape C:/Data/microblaze.svg
```

Send Feedback

See Also

- create_bd_design
- current_bd_design
- open_bd_design
- regenerate_bd_layout
- write_schematic

write_bd_tcl

Export the current design to a Tcl file on disk.

Syntax

```
write_bd_tcl [-force] [-bd_name <arg>] [-no_mig_contents] [-no_ip_version]
    [-ignore_minor_versions] [-bd_folder <arg>] [-check_ips <arg>]
    [-hier_blks <arg>] [-include_layout] [-exclude_layout] [-make_local]
    [-no_project_wrapper] [-exclude_pfm] [-updated_pfm_attrs] [-quiet]
    [-verbose] <tcl_filename>
```

Returns

TCL_OK, TCL_ERROR if failed.

Usage

Name	Description
[-force]	Flag to overwrite existing file.
[-bd_name]	Name for block diagram. By default will use current block diagram's name.
[-no_mig_contents]	Flag to not include MIG PRJ contents into generated Tcl script, but instead will load PRJ from working directory. Default is to include MIG PRJ contents in Tcl script.
[-no_ip_version]	Flag to not include the IP version as part of the IP VLNV in create_bd_cell commands. NOTE - this may have implications if there are major IP version changes.
[-ignore_minor_versions]	Use this flag to create the cells in the design using their latest minor version. For example, a project contains versions of blk_mem_gen IP like 7.3, 7.4, 8.3, 8.4. In the design there is a blk_mem_gen_v7.4. With this flag, write_bd_tcl will generate the line: create_bd_cell -type ip -vlnv xilinx.com:ip:blk_mem_gen:7.* bmg_0_v7. When the generated Tcl script is sourced, the cell bmg_0_v7 will use the latest blk_mem_gen_v7.
[-bd_folder]	Remote BD feature - Specify the folder where the design will be generated when Tcl script is sourced.
[-check_ips]	By default value = true, therefore, will check if IPs/modules exist in the IP catalog or project before continuing to reconstruct the design. Valid values are (true/false), (yes/no), or (1/0).
[-hier_blks]	Comma separated list of hierarchical blocks in the design that will be generated by the Tcl script. Will include any sub-hierachical blocks within the specified blocks too. This option will not create the top-level design portion.
[-include_layout]	By default will NOT include the GUI layout of the design. Use this argument to include the layout information in the generated Tcl script.
[-exclude_layout]	NOTE - This flag will be obsolete in a near future release, but is currently supported for backwards compatibility. Use this argument to not include the layout information in the generated Tcl script.

Send Feedback

Name	Description
`[-make_local]`	Use this flag when you want to write your remote BD out as a local BD.
`[-no_project_wrapper]`	This option is used to write the BD create TCL procs without any project wrapper.
`[-exclude_pfm]`	Use this flag to exclude pfm attributes for this design
`[-updated_pfm_attrs]`	Use this flag to update the pfms to remove connected ports/ interfaces from the values.
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<tcl_filename>`	Name of exported Tcl file

Categories

[IPIntegrator](#)

Description

Export the current IP integrator subsystem design as a Tcl script file to the disk.

> **IMPORTANT!** *Any directory in the path specified by the* `<name>` *option must already exist, or the script will not be created.*

The Tcl script file lets you recreate, reuse, and customize IP integrator subsystem designs without having to archive the original subsystem design.

When working with a new software release, you must use the output script from the `write_bd_tcl` command to create a block design in the same software release as the Tcl script was generated. This ensures the availability of the needed versions of any IP used in the script. You can then migrate the created block design into a new software release.

This command returns TCL_OK if it is successful, or TCL_ERROR if it fails, unless `-quiet` is specified.

Arguments

`-force` - (Optional) Overwrite an existing bd_tcl file of the same name if it already exists.

`-bd_name <arg>` - (Optional) Specify the name to assign to the block diagram in the bd_tcl file. When the Tcl script is run, the new block diagram will be created with the specified name. By default, the current name of the block diagram will be used.

`-no_mig_contents` - (Optional) Do not include memory IP PRJ contents into the generated Tcl script. By default this content will be included in the Tcl script.

`-no_ip_version` - (Optional) Do not include the Version in the Vendor:Library:Name:Version (VLNV) value that specifies the IP for `create_bd_cell` commands in the `write_bd_tcl` file. This allows a bd_tcl script to create a new block diagram using the latest version of the IP from the Vivado IP catalog.

Note: This can have significant design implications when IP used in a block design have undergone major version changes from when the bd_tcl file was written to when it is used.

`-ignore_minor_versions` - (Optional) Do not include the minor version in the IP VLNV. This lets the bd_tcl script use the latest minor revision of an IP, while preserving the major version. For example, an IP blk_mem_gen_v7.4 will be written as `-vlnv xilinx.com:ip:blk_mem_gen:7.*`.

`-bd_folder <arg>` - (Optional) Specify the directory where the block diagram will be generated when the bd_tcl script is run. This lets you specify the block design to be created outside of the directory structure of a project where the bd_tcl script is being run.

`-check_ips [ true | false ]` - (Optional) A boolean argument that adds a check to the Tcl script to ensure that all required IP are found before starting the process of creating the block diagram when the script is run. If any IP are missing, the script will error out rather than attempting to create the block design. The default value is `true`.

`-hier_blks <arg>` - (Optional) Only generate the `write_bd_tcl` script for specified hierarchical blocks or modules of the block design.

> **TIP:** *Hierarchical block modules must be specified as bd_cell objects as returned by `get_bd_cells` for example. In addition, if the specified bd_cells are not hierarchical blocks, then no Tcl script will be generated.*

`-include_layout` - (Optional) Include the graphical layout data of the block design in the output Tcl script. This preserves the current layout of the block design in the Vivado IP integrator design canvas in the output Tcl script. This option is ignored when used with the `-hier_blks` option, as the layout information is written from the top-level of the block design.

`-make_local` - (Optional) Specify this option when the block diagram is remote from the current project, but you want to make it local to the project when running the script to recreate it.

`-no_project_wrapper` - (Optional) Write a Tcl script to recreate the block diagram without creating the top-level wrapper for the design.

`-exclude_pfm` - (Optional) Use this flag to exclude PFM properties used in defining a Vitis platform from the created bd_tcl script.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<tcl_filename> - Specify the name of the Tcl file to write. A suffix of `.tcl` will be supplied if no file suffix is specified.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

Example

The following example creates a Tcl script from the current IP integrator subsystem design:

```
write_bd_tcl C:/Data/myDesign.tcl
```

This example creates a Tcl script from the current IP integrator subsystem design, specifies a new name for the block diagram, specifies that IP versions should not be included, and also indicates a folder to write the block diagram to when it is created by running the output bd_tcl script:

```
write_bd_tcl -bd_name newMB1 -no_ip_version \
    -bd_folder C:/Block_Designs projMB1
```

Note: The file suffix of `.tcl` will be appended to the specified file name, resulting in a file name of `projMB1.tcl`.

The following example creates a Tcl script for the specified hierarchical block cell, overwriting any existing script of the same name:

```
write_bd_tcl -force -hier_blks [get_bd_cells myHier] \
-include_layout C:/Data/myHier.tcl
```

Note: The `-include_layout` option is ignored in this example, due to the use of the `-hier_blks` option.

See Also

- create_bd_design
- open_bd_design
- save_bd_design

write_bitstream

Write a bitstream for the current design

Syntax

```
write_bitstream [-force] [-verbose] [-raw_bitfile] [-no_binary_bitfile]
    [-mask_file] [-readback_file] [-logic_location_file] [-bin_file]
    [-reference_bitfile <arg>] [-cell <arg>] [-no_partial_bitfile] [-quiet]
    <file>
```

Usage

Name	Description
`[-force]`	Overwrite existing file
`[-verbose]`	Print write_bitstream options
`[-raw_bitfile]`	Write raw bit file (.rbt)
`[-no_binary_bitfile]`	Do not write binary bit file (.bit)
`[-mask_file]`	Write mask file (.msk)
`[-readback_file]`	Write readback files (.rbd, .msd)
`[-logic_location_file]`	Write logic location file (.ll)
`[-bin_file]`	Write binary bit file without header (.bin)
`[-reference_bitfile]`	Reference bitfile to be used for generating partial bitstream
`[-cell]`	Create only partial bitstream for named cell
`[-no_partial_bitfile]`	Do not write partial bit files for a Dynamic Function eXchange design
`[-quiet]`	Ignore command errors
`<file>`	The name of the .bit file to generate

Categories

FileIO

Description

Writes a bitstream file for the current project. This command must be run on an Implemented Design. The bitstream written will be based on the open Implemented Design.

The files that can be generated by the `write_bitstream` command include the following:

- Bit file: The binary bitstream file (`.bit`).

- Raw (ASCII) Bit file: A raw bit file (`.rbt`) that contains the same information as the binary bitstream file, but is in ASCII format.

- Mask file: A mask file (`.msk`) that has mask data in place of the configuration data in the bitstream file.

- Logic Location file: An ASCII logic location file (`.ll`) that shows the bitstream position of latches, flip-flops, LUTs, Block RAMs, and I/O block inputs and outputs.

- Bin file: A binary file (`.bin`) containing only the device programming data, without the header information found in the standard binary Bit file.

- Reference Bit file: An incremental bitstream file containing only the differences from the current bitstream and a specified reference bitstream.

The Vivado tool can write a compressed bitstream, if you have enabled compression by setting the BITSTREAM.GENERAL.COMPRESS property on the implemented design. Refer to the *Vivado Design Suite User Guide: Programming and Debugging* (UG908) for more information on Device Configuration Properties. To enable compression use the following Tcl command:

```
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
```

The Vivado Design Suite can also write an encrypted bitstream to protect the intellectual property of the design in the bitstream. To create an encrypted bitstream you must first define the type of encryption to be used, and the encryption key. You can accomplish this most easily using the Encryption page of the Edit Device Properties dialog box in the Vivado IDE. Refer to the *Vivado Design Suite User Guide: Programming and Debugging* (UG908) for more information on the Edit Device Properties dialog box.

You can also enable encryption by manually defining the appropriate properties on the implemented design as follows:

```
set_property BITSTREAM.ENCRYPTION.ENCRYPT YES [get_designs impl_1]
set_property BITSTREAM.ENCRYPTION.ENCRYPTKEYSELECT EFUSE [get_designs impl_1]
set_property BITSTREAM.ENCRYPTION.KEY0 8675309 [get_designs impl_1]
```

The properties associated with encryption include:

- BITSTREAM.ENCRYPTION.ENCRYPT - Enables encryption when generating the bitstream with `write_bitstream`. This property accepts a value of YES or NO.

- BITSTREAM.ENCRYPTION.ENCRYPTKEYSELECT - Specifies the method for storing the encryption key on the hardware device. The accepted values are BBRAM and EFUSE, referring to battery-backed SRAM or the eFUSE registers on the device.

> ⚠ **CAUTION!** *eFUSEs are one-time programmable cells on the hardware device, used to store the factory-programmed Device DNA, AES-GCM encryption key, and user specified values. Refer to the UltraScale Architecture Configuration User Guide (UG570) or 7 Series FPGAs Configuration User Guide (UG470) for more information on eFUSE registers.*

- BITSTREAM.ENCRYPTION.KEY0 - Specifies the encryption key to apply to the BBRAM, or the eFUSE FUSE_KEY registers on the device. The key can be specified as a 256 bit value, and will be required when accessing an encrypted bitstream to program, verify, or readback the hw_device.

> 💡 **TIP:** *The `write_bitstream` command will write an NKY file of the same name as the bitstream file (with the `.nky` extension) when the BITSTREAM.ENCRYPTION.KEY0 property is specified. This encryption file can then be used in other designs by setting the BITSTREAM.ENCRYPTION.KEYFILE property.*

- BITSTREAM.ENCRYPTION.KEYFILE - Specifies an encryption key file (NKY or NKZ) as an alternative to setting the ENCRYPTION.KEY0 property. The specified encryption key file will be used during bitstream encryption.

> ⭐ **IMPORTANT!** *If both the BITSTREAM.ENCRYPTION.KEY0 and BITSTREAM.ENCRYPTION.KEYFILE properties are defined, the tool will use the encryption key specified by the BITSTREAM.ENCRYPTION.KEY0 property and return a message to that effect.*

Arguments

`-force` - (Optional) Force the overwrite of an existing bitstream file of the same name.

`-verbose` - (Optional) Print details of the options applied to the bitstream when running the `write_bitsream` command.

`-raw_bitfile` - (Optional) Write a raw bit file (`.rbt`) which contains the same information as the binary bitstream file, but is in ASCII format. The output file will be named `<file>.rbt`.

`-no_binary_bitfile` - (Optional) Do not write the binary bitstream file (`.bit`). Use this command when you want to generate the ASCII bitstream or mask file, or to generate a bitstream report, without also generating the binary bitstream file.

`-mask_file` - (Optional) Write a mask file (`.msk`), which has mask data where the configuration data is in the bitstream file. This file determines which bits in the bitstream should be compared to readback data for verification purposes. If a mask bit is 0, that bit should be verified against the bitstream data. If a mask bit is 1, that bit should not be verified. The output file will be named `<file>.msk`.

`-readback_file` - (Optional) Lets you perform the Readback function by creating the necessary readback files (`.rbd`, `.msd`).

- `.rbd` - An ASCII file that contains only expected readback data, including pad words and frames. No commands are included.

- `.msd` - An ASCII file that contains only mask information for verification, including pad words and frames. No commands are included.

`-logic_location_file` - (Optional) Creates an ASCII logic location file (.ll) that shows the bitstream position of latches, flip-flops, LUTs, Block RAMs, and I/O block inputs and outputs. Bits are referenced by frame and bit number in the location file to help you observe the contents of FPGA registers.

`-bin_file` - (Optional) Creates a binary file (`.bin`) containing only device programming data, without the header information found in the standard bitstream file (`.bit`).

`-reference_bitfile <arg>` - (Optional) Read a reference bitstream file, and output an incremental bitstream file containing only the differences from the specified reference file. This partial bitstream file can be used for incrementally programming an existing device with an updated design.

`-cell <arg>` - (Optional) Write a partial bitstream file for the specified cell or block level of the design hierarchy. The bitstream file will only include programming data for the specified cell or module.

`-no_partial_bitfile` - (Optional) Do not output a partial bit file for a Partial Reconfiguration module or design. Refer to the *Vivado Design Suite User Guide: Dynamic Function eXchange* (UG909) for more information on the PR flow.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`<file>` - (Required) The name of the bitstream file (`.bit`) to write. If you do not specify a file extension, the `.bit` extension will be added by the tool, but you cannot specify an extension other than `.bit`.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

Examples

The following example enables compression and writes a bitstream file of the specified name:

```
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
write_bitstream design1.bit
```

The following example writes both the binary and ASCII forms of the bitstream:

```
write_bitstream -raw_bitfile C:/Data/design1
```

Note: The appropriate file extension will be added by the tool.

See Also

- create_hw_bitstream
- launch_runs
- program_hw_devices
- readback_hw_device

- verify_hw_devices

Send Feedback

write_bmm

Write a bmm file

Syntax

```
write_bmm [-force] [-quiet] [-verbose] <file>
```

Returns

The name of the bmm file.

Usage

Name	Description
[-force]	Overwrite existing BMM file.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<file>	Design bmm file Values: A filename with alphanumeric characters and .bmm extension.

Categories

FileIO

Description

The Block RAM Memory Map (BMM) file is a text file that describes how individual block RAMs on an FPGA are grouped together into a contiguous address space called an Address Block.

The `write_bmm` command exports BMM information from the current design to the specified file. For implemented designs the BMM file will be include placement information. The `data2mem` command uses the BMM file as input to direct the translation of programming data into the proper form for use in simulation, device programming, or software development in SDK.

The command returns the name of the output file, or an error.

Arguments

`-force` - (Optional) Overwrite the BMM file if it already exists.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file>` - (Required) The filename of the BMM file to write.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

Example

The following example writes the BMM file for the current design:

```
write_bmm C:/Data/design1.bmm
```

Send Feedback

write_bsdl

Generate a design specific post-configuration BSDL file (.bsd).

Syntax

```
write_bsdl [-force] [-bsd <arg>] [-quiet] [-verbose] <file>
```

Returns

Name of the output file.

Usage

Name	Description
[-force]	Overwrite existing .bsd file
[-bsd]	Specify an updated generic BSDL file.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<file>	Output file name. The .bsd extension is optional.

Categories

FileIO

Description

Generate a Boundary Scan Description Language (BSDL) file (.bsd) for the current design that reflects the post-configuration boundary scan architecture of the target device.

The boundary scan architecture for the device is changed when the device is configured because certain connections between the boundary scan registers and pad may change. These changes must be communicated to the boundary scan tester through a post-configuration BSDL file. Refer to the *Vivado Design Suite User Guide: Programming and Debugging (UG908)* for more information on the available configuration modes.

The `write_bsdl` command reads a pre-configuration BSDL file for the target part from the Vivado Design Suite installation area, and combines that with post-configuration data from the current design.

This command returns the name of the output BSDL file, or returns an error if it fails.

Arguments

`-force` - (Optional) Overwrite an existing BSDL file of the same name.

Send Feedback

`-bsd <arg>` - (Optional) Specify an existing BSDL file to update. Use this to update a generic BSDL file from the Vivado Design Suite installation with post-configuration data from the current design.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file>` - (Required) Specify the output file name. The `.bsd` extension is optional.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

Example

The following example writes a BSDL file at the specified location:

```
write_bsdl -force C:/Data/project/design1.bsd
```

See Also

- write_bitstream

write_cfgmem

Create file(s) for programming flash memory.

Syntax

```
write_cfgmem [-force] -format <arg> -size <arg> [-interface <arg>]
    [-checksum] [-disablebitswap] [-loadbit <arg>] [-loaddata <arg>]
    [-quiet] [-verbose] <file>
```

Usage

Name	Description
[-force]	Overwrite existing file
-format	Format of the file to generate
-size	Size of memory that is being targeted in M Bytes (must be power of 2).
[-interface]	Interface used to program device. Default: SMAPx8
[-checksum]	Calculate a 32-bit checksum for each file. Memory will be filed with value of 0xFF unless a different byte value is specified. Default: 0xFF
[-disablebitswap]	Disable bit swapping in a byte for bitfiles.
[-loadbit]	Load bit files into memory from given address.
[-loaddata]	Load data into memory from given address.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<file>	The name of the file to generate

Categories

FileIO

Description

This command formats a design specific configuration bitstream (`.bit`) file, and any specified data files, into a specified memory configuration file format to program into a flash memory device using the `program_hw_cfgmem` command. Supported memory configuration file formats are MCS, BIN, and HEX.

> 💡 **TIP:** *When you generate a cfgmem file with* `write_cfgmem`, *by default the bits within a byte are bit-swapped, or mirrored, compared to bytes in the original input bitstream. You can disable bitswap using the* `-disablebitswap` *option as described below.*

The process whereby the design specific data is loaded or programmed into the Xilinx® FPGA is called configuration. The `create_hw_cfgmem` command defines a flash memory device used for configuring and booting the hardware device.

Send Feedback

After the hw_cfgmem object is created, and associated with the hw_device, the configuration memory can be programmed with the bitstream and other data from a memory configuration file created with the `write_cfgmem` command. The hw_cfgmem object is programmed using the `program_hw_cfgmem` command.

The `write_cfgmem -loadbit` command loads one or more specified bitstream files into the memory configuration file, filling the available memory of the device in an upward or downward direction from a specified starting address. You can also add data files to the memory configuration file, by specifying the starting address to load the file with `-loaddata`.

> 💡 **TIP:** *When using `-loadbit` and `-loaddata` to fill the memory of the device, you must exercise care to insure that the bitstream and data files fit into the available memory and do not overwrite each other. Any data collisions will cause the `write_cfgmem` command to fail with an error.*

The `write_cfgmem` command returns a transcript of its process when successful, or returns an error if it fails.

Arguments

`-force` - (Optional) Overwrite a file of the same name if one exists.

`-format [ BIN | HEX | MCS ]` - (Required) The format of the memory configuration file to write. Supported values include BIN, HEX, and MCS.

`-size <arg>` - (Required) Specify the size limit in MBytes of the PROM device that is being targeted. The size must be specified as a power of 2.

`-interface <arg>` - (Optional) Specify the interface used to program the PROM device. Valid values include SMAPx8 (default), SMAPx16, SMAPx32, SERIALx1, SPIx1, SPIx2, SPIx4, SPIx8, BPIx8, BPIx16. This also determines if byte swapping is enabled or disabled. The default interface is SMAPx8.

> ⭐ **IMPORTANT!** *The specified interface format of the configuration memory file is critical to properly programming the flash memory device with the `program_hw_cfgmem` command. You should be careful to use this option to match the generated file with the target cfgmem_part.*

`-checksum` - (Optional) Calculate a 32-bit checksum for the PROM file. The device memory will be filled with the default value of 0xFF unless a different byte value is specified. This option generates a checksum value appearing in the memory configuration file. This value should match the checksum in the device programmer. Use this option to verify that correct data was programmed into the flash memory.

`-disablebitswap` - (Optional) Disable the default bit swapping for bytes in the bitstream files. By default, in the files written by `write_cfgmem`, the bits within a byte are bit-swapped, or mirrored, compared to bytes in the original input BIT files. This option disables the bit swapping in the output files.

`-loadbit <arg>` - (Optional) Specify the starting address of the PROM device to begin loading one or more bitstream files. The option is specified as a string with the form:

```
"up/down 0x0 <bitfile1>.bit <bitfile2>.bit"
```

Where:

- `up | down` - This option loads one or more BIT files into memory, starting from the specified address, in either and upward or a downward direction.

- `0x0` - The starting address to load the bitstream, specified as a hexadecimal value.

- `<bitfile>.bit` - The bitstream (`.bit`) file to load into the flash memory device. You can specify multiple bitstream files, causing the files to be concatenated in a daisy chain.

> **TIP:** *You can only specify the* `-loadbit` *option once, but you can repeat the arguments as needed to load multiple bitstream files from different starting addresses:*
>
> ```
> -loadbit "up 0 bitfile1.bit up 0xFFFFFF bitfile2.bit"
> ```

`-loaddata <arg>` - (Optional) Load the specified data files into the memory of the configuration device from the starting address. The `-loaddata` option is a string in the same form as the `-loadbit` argument, specifying the direction, starting address, and data file names to add into the memory configuration file. Data files will be added to the flash memory device as is, with no additional formatting.

- `up | down` - This option loads one or more DATA files into memory, starting from the specified address, in either and upward or a downward direction.

- `0x0` - The starting address to load the data file, specified as a hexadecimal value.

- <data_file> - A data file to load into the flash memory device. You can specify multiple data files, causing the files to be concatenated in a daisy chain.

Note: Although both `-loadbit` and `-loaddata` are marked as optional, at least one argument must be used to provide the data for the memory configuration file.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file>` - (Required) The filename of the memory configuration file to write. The file extension will match the format specified (`.mcs`), and is not required as part of the file name.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

Example

The following example writes the specified memory configuration file in the MCS format, with a size limit of 64 MB, loading the specified bitstream file moving up from the starting address:

```
write_cfgmem -format MCS -size 64 -loadbit "up 0x0 \
C:/Data/Vivado_Debug/project_debug/project_debug.runs/impl_1/
sinegen_demo.bit" \
config_memory1
```

See Also

- create_hw_cfgmem
- current_hw_device
- delete_hw_cfgmem
- get_cfgmem_parts
- get_property
- program_hw_cfgmem
- readback_hw_cfgmem
- set_property
- write_bitstream

write_checkpoint

Write a checkpoint of the current design

Syntax

```
write_checkpoint [-force] [-cell <arg>] [-logic_function_stripped]
    [-encrypt] [-key <arg>] [-incremental_synth] [-quiet] [-verbose]
    [<file>]
```

Returns

The name of the checkpoint file.

Usage

Name	Description
[-force]	Overwrite existing checkpoint file
[-cell]	Write a checkpoint of this cell
[-logic_function_stripped]	Convert INIT strings on LUTs & RAMBs to fixed values. Note that the resulting netlist will be nonfunctional.
[-encrypt]	Encrypt unprotected modules using IEEE 1735 IP security version 2
[-key]	Key file to be used with -encrypt option; Otherwise, use Xilinx public key
[-incremental_synth]	export synthesis archive file to be used for re-using implementation.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<file>]	Design checkpoint file Values: A filename with alphanumeric characters and .dcp extension.

Categories

[FileIO](#)

Description

Saves the design at any point in the design process so that you can quickly import it back into the tool as needed. A design checkpoint (DCP) can contain the netlist, the constraints, and any placement and routing information from the implemented design.

> 💡 **TIP:** *In the Project mode, a DCP will not have timing constraints after synthesis. The timing constraints are annotated against the design during* `open_run` *or* `link_design` *commands, or when launching an implementation run. To create a DCP with timing constraints, create the design checkpoint after* `opt_design`*, or after the implementation run completes.*

Use the `read_checkpoint` command to import a checkpoint file.

Arguments

`-force` - (Optional) Overwrite an existing checkpoint file of the same name if it already exists.

`-cell <arg>` - (Optional) Instructs the tool to output the contents of the specified hierarchical cell into a checkpoint file. Only one cell can be specified for output.

`-logic_function_stripped` - (Optional) Hides the INIT values for LUTs & RAMs by converting them to fixed values in order to create a checkpoint for debug purposes that will not behave properly in simulation or synthesis.

`-encrypt` - (Optional) Encrypt the checkpoint netlist.

`-key <arg>` - (Optional) Specify the key to use in encrypting and decrypting the checkpoint. If not specified, the Xilinx public key is used by default.

`-incremental_synth` - (Optional) Enable the synthesis checkpoint to be used for incremental synthesis. The checkpoint must be enabled for incremental synthesis to be used for subsequent incremental synthesis runs.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file>` - (Required) The name of the checkpoint file to be created. A `.dcp` extension will be added if no extension is specified.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

Examples

The following example creates the specified checkpoint file, overwriting a file of the same name if one already exists:

```
write_checkpoint C:/Data/checkpoint1 -force
```

Note: The tool will add the `.dcp` extension to the specified file name, and will overwrite an existing `checkpoint1.dcp` file.

Send Feedback

See Also

- read_checkpoint

write_csv

Export package pin and port placement information

Syntax

```
write_csv [-force] [-quiet] [-verbose] <file>
```

Returns

Name of the output file.

Usage

Name	Description
[-force]	Overwrite existing file
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<file>	Pin Planning CSV file

Categories

FileIO

Description

Writes package pin and port placement information into a comma separated value (CSV) file.

The specific format and requirements of the CSV file are described in the *Vivado Design Suite User Guide: I/O and Clock Planning (UG899)*.

Arguments

`-force` - (Optional) Overwrite the CSV file if it already exists.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file>` - (Required) The filename of the CSV file to write.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

Examples

The following example exports a CSV file from the current project:

```
write_csv C:/Data/pinList.csv
```

See Also

- read_csv

Send Feedback

write_debug_probes

Write debug probes to a file

Syntax

```
write_debug_probes [-cell <arg>] [-no_partial_ltxfile] [-force] [-quiet]
    [-verbose] <file>
```

Returns

Name of the output file.

Usage

Name	Description
[-cell]	Hierarchical name of the Reconfigurable Partition Cell
[-no_partial_ltxfile]	Do not generate partial LTX files
[-force]	Overwrite existing file
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<file>	Debug probes file name (default extension is .ltx)

Categories

FileIO, Debug

Description

Writes a Vivado Design Suite logic analyzer probes file containing ILA debug cores and signal probes added to the current design. The debug probes data file typically has a `.ltx` file extension.

ILA cores are added to the design using the `create_debug_core` command. ILA probes are added to the design using the `create_debug_port` command, and connected to nets in your design using the `connect_debug_port` command.

The specific information and use of the debug probes file is described in the *Vivado Design Suite User Guide: Vivado Programming and Debugging (UG908)*.

Arguments

`-cell <arg>` - (Optional) Specify the hierarchical name of a reconfigurable partition cell to export a partial probe file.

-force - (Optional) Overwrite the specified file if it already exists.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the set_msg_config command.

<file> - (Required) The file name of the debug probes file to write.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

Examples

The following example write a debug probe file from the current design:

```
write_debug_probes C:/Data/designProbes.ltx
```

See Also

- create_debug_core
- connect_debug_cores
- implement_debug_core

write_device_image

Write programmable device image

Syntax

```
write_device_image [-force] [-verbose] [-raw_partitions] [-mask_file]
    [-logic_location_file] [-cell <arg>] [-no_pdi] [-no_partial_pdifile]
    [-quiet] <file>
```

Usage

Name	Description
[-force]	Overwrite existing file
[-verbose]	Print write_device_image options
[-raw_partitions]	Write raw CFI and NPI partition files (.rcfi and .rnpi)
[-mask_file]	Write mask file (.msk_cfi)
[-logic_location_file]	Write logic location file (.ll)
[-cell]	Create only partial device image for named cell
[-no_pdi]	Do not generate pdi file
[-no_partial_pdifile]	Do not write partial pdi files for a Dynamic Function eXchange design
[-quiet]	Ignore command errors
<file>	The name of the .rcfi and .rnpi files to generate

Categories

[FileIO](#)

write_edif

Export the current netlist as an EDIF file

Syntax

```
write_edif [-pblocks <args>] [-cell <arg>] [-force] [-security_mode <arg>]
     [-logic_function_stripped] [-quiet] [-verbose] <file>
```

Returns

The name of the output file or directory.

Usage

Name	Description
`[-pblocks]`	Export netlist for these pblocks (not valid with -cell)
`[-cell]`	Export netlist for this cell (not valid with -pblocks)
`[-force]`	Overwrite existing file
`[-security_mode]`	If set to 'all', and some of design needs encryption then whole of design will be written to a single encrypted file Default: multifile
`[-logic_function_stripped]`	Convert INIT strings on LUTs & RAMBs to fixed values
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<file>`	Output file (directory with -pblocks or -cell)

Categories

FileIO

Description

Writes the current netlist as an EDIF file, or outputs the contents of specific Pblocks or hierarchical cells as EDIF netlist files.

In the case of either the `-pblocks` or `-cell` option being used, this argument specifies a directory name where the EDIF netlist files for each Pblock or cell will be written. The EDIF netlist file will be named after the Pblock or cell. If the directory specified does not exist, the tool will return an error.

Send Feedback

Arguments

`-pblocks <arg>` - (Optional) Instructs the tool to output the contents of the specified Pblocks as EDIF netlist files. The contents of each Pblock will be written to a separate EDIF file. These files can be added as design source files to netlist projects, but are not intended to be read into a design using `update_design`. Use the `-cell` option to write EDIF netlists for use with `update_design`.

`-cell <arg>` - (Optional) Instructs the tool to output the contents of the specified hierarchical cell as EDIF netlist files. Only one cell can be specified for output.

Note: The `-pblocks` and `-cell` options are mutually exclusive. Although they are optional arguments, only one can be specified at one time.

`-force` - (Optional) Overwrite the EDIF file if it already exists.

`-security_mode [ multifile | all ]` - (Optional) Write a multiple EDIF files when encrypted IP is found in the design, or write a single file.

- `multifile` - This is the default setting. By default the command writes out the full design netlist to the specified file. However, if the design contains secured IP, it creates an encrypted file containing the contents of the secured module. This will result in the output of multiple EDIF files, containing secured and unsecured elements of the design.

- `all` - Write both encrypted and unencrypted cells to a single specified file.

`-logic_function_stripped` - (Optional) Hides the INIT values for LUTs & RAMs by converting them to fixed values in order to create a netlist for debug purposes that will not behave properly in simulation or synthesis.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file>` - (Required) The filename of the EDIF file to write. The default file extension for an EDIF netlist is .edn. If the `-pblocks` or `-cell` options are used, the name specified here refers to a directory rather than a single file.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

Send Feedback

Examples

The following example writes an EDIF netlist file for the whole design to the specified file name:

```
write_edif C:/Data/edifOut.edn
```

The following example outputs an EDIF netlist for all Pblocks in the design. The files will be written to the specified directory.

```
write_edif -pblocks [get_pblocks] C:/Data/FPGA_Design/
```

See Also

- read_edif

write_hw_ila_data

Write hardware ILA data to a file

Syntax

```
write_hw_ila_data [-force] [-csv_file] [-vcd_file] [-legacy_csv_file]
    [-quiet] [-verbose] <file> [<hw_ila_data>] [<hw_ila_data>]
```

Returns

Name of the output file.

Usage

Name	Description
[-force]	Overwrite existing file
[-csv_file]	Export CSV format file only
[-vcd_file]	Export VCD format file only
[-legacy_csv_file]	Export csv file without radix information.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<file>	hardware ILA data file name
[<hw_ila_data>]	hardware ILA data object Default: Current hardware ILA data

Categories

Hardware

Description

Write the ILA debug core sample data, stored in the specified hw_ila_data object, to a binary file on the disk.

A hw_ila_data object is created when the hw_ila is triggered on the hw_device, or by the `upload_hw_ila_data` command when moving the captured data from the physical FPGA device, hw_device.

The `write_hw_ila_data` lets you write the data of the hw_ila_data object to a binary file on the disk for later review. You can read the ILA debug core data back into the Vivado logic analyzer using the `read_hw_ila_data` command, which creates a new hw_ila_data object.

This command returns the name of the file written, or returns an error if it fails.

Send Feedback

Arguments

-force - (Optional) Overwrite an existing file of the same name if one exists.

-csv_file - (Optional) Export a comma-separated values (CSV) file only. This configures the write_hw_ila_data command to export the ILA data in the form of a CSV file that can be used to import into a spreadsheet or third-party application, rather than the default binary ILA file format.

-vcd_file - (Optional) Export a value change dump (VCD) file only. This configures the write_hw_ila_data command to export the ILA data in the form of a VCD file that can be used to import into a third-party application or viewer, rather than the default binary ILA file format.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the set_msg_config command.

<file> - (Required) The filename of the ILA data file to write. The default file extension for an ILA data file is .ila. The default file extension for -csv_file is .csv, and for -vcd_file is .vcd.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

<hw_ila_data> - (Optional) The hardware ILA data to write to the specified file. The hw_ila_data must be specified as an object, as returned by the get_hw_ila_datas or the current_hw_ila_data commands. If no hw_ila_data object is specified, the current hw_ila_data is written to the specified file.

Example

The following example uploads the data from the hw_ila debug core into a hw_ila_data object, and then writes that data object to the specified ILA data file, overwriting an existing file if one exists:

```
write_hw_ila_data -force design1_ila_data [upload_hw_ila_data hw_ila_1]
```

This example triggers the hw_ila, then writes the captured hw_ila_data to a CSV file:

```
run_hw_ila hw_ila_1
write_hw_ila_data -csv_file C:/Data/design1_ila_data [current_hw_ila_data]
```

See Also

- current_hw_ila
- current_hw_ila_data
- get_hw_ilas
- get_hw_ila_datas
- run_hw_ila
- upload_hw_ila_data
- wait_on_hw_ila

write_hw_platform

Write the Xilinx Shell Archive for the current design

Syntax

```
write_hw_platform [-fixed] [-force] [-include_bit] [-include_sim_content]
    [-minimal] [-hw] [-hw_emu] [-quiet] [-verbose] [<file>]
```

Returns

The name of the Shell file.

Usage

Name	Description
[-fixed]	Write fixed Shell.
[-force]	Overwrite existing Xilinx Shell Archive file
[-include_bit]	Include bit file(s) in the Shell.
[-include_sim_content]	Include simulaton files in the Shell.
[-minimal]	Add only minimal files in the Shell.
[-hw]	Write shell for use in hardware only.
[-hw_emu]	Write shell for use in hardware emulation only.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<file>]	Device Support Archive file Values: A filename with alphanumeric characters and .xsa extension.

Categories

[Vitis](#), [Platform](#), [FileIO](#)

Description

Writes a Xilinx support archive (XSA) of the current design for use as a hardware platform.

All platforms are dynamically implemented during compilation, meaning that the accelerator logic coming is implemented along with some or all of the logic contained in the hardware platform design. The XSA represents the required portion of the hardware platform. The hardware platform design used to create a XSA consists of a Vivado IP integrator subsystem design with all the required board interface IP cores configured and connected to the device I/Os. The Vivado project must also include several required XSA and PFM properties needed to define the XSA.

This command reports the name of the XSA file written, or returns an error if it fails.

Arguments

`-fixed` - (Optional) Overwrite a fixed shell XSA, which enables the platform for use in the software development flow, but not for use in acceleration.

`-force` - (Optional) Overwrite the XSA file if it already exists.

`-include_bit` - (Optional) Include the bitstream for the current design in the XSA. By default MCS files are created by `write_hw_platform`, and the bitstreams are discarded. When this options is specified, bitstream files are preserved for use in debugging the platform.

`-include_emulation` - (Optional) This option exports data needed to support hardware emulation at the same time that the XSA is created. This option requires special features of the hardware platform, and is not generally recommended.

`-minimal` - (Optional) Add only those files in the XSA that are required to run the Vitis flow. Do not add anything extra. When this option is specified the `write_hw_platform` command ignores other options that add content to the XSA, such as the `-include_bit` and `-include_emulation` options.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file>` - (Required) The name of the XSA file to write.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

Examples

The following example writes the XSA for the current project, overwriting a XSA file of the same name if one exists:

```
write_hw_platform -force C:/Data/zc702.xsa
```

See Also

- open_hw_platform
- validate_hw_platform

write_hw_platform_metadata

Write the unified JSON metadata file for the current design

Syntax

```
write_hw_platform_metadata [-quiet] [-verbose] [<file>]
```

Returns

The name of the unified JSON metadata file.

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<file>]	Unified JSON metadata file Values: A filename with alphanumeric characters and .json extension.

Categories

Vitis, Platform, FileIO

Description

Writes a JSON metadata file for the platform represented by the current design.

This command returns the name of the JSON file written, or returns an error if it fails.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file>` - (Required) The name of the JSON file to write. Must use the .json file extension.

Send Feedback

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

Examples

The following example writes the JSON file for the current project:

```
write_hw_platform_metadata C:/Data/zc102_platform.json
```

See Also

- open_hw_platform
- validate_hw_platform
- write_hw_platform

Send Feedback

write_hw_sio_scan

Write scan data to a file.

Syntax

```
write_hw_sio_scan [-force] [-quiet] [-verbose] <file> <hw_sio_scan>
```

Returns

Name of the output file.

Usage

Name	Description
[-force]	Overwrite existing file
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<file>	hardware SIO_scan file name
<hw_sio_scan>	hardware SIO scan data object

Categories

Hardware

Description

Write the populated hw_sio_scan object after `run_hw_sio_scan` completes.

To analyze the margin of a given link, it is often helpful to run a scan of the link using the specialized Eye Scan hardware of Xilinx UltraScale devices or 7 series FPGAs. The Vivado serial I/O analyzer feature lets you to create, run, and save link scans.

This command saves the scan to disk after completing the scan run. The format of the file is a CSV file of values observed while running the scan.

This command returns the filename of the file output, or returns an error if the command fails.

Arguments

`-force` - (Optional) Overwrite the specified file if it already exists.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file>` - (Required) The filename of the scan file to write. The default suffix of `.csv` will be assigned to the scan file if a suffix is not specified.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

<hw_sio_scan> - (Required) Specify a hw_sio_scan object to write to disk. The hw_sio_scan must be specified as an object as returned by the `get_hw_sio_scans` command.

Example

The following example writes the specified hw_sio_scan object to disk, over writing a file of the same name if one exists:

```
write_hw_sio_scan -force C:/Data/Vivado_Debug/LoopBack_1.csv \
[get_hw_sio_scans {SCAN_3}]
```

See Also

- create_hw_sio_scan
- create_hw_sio_sweep
- get_hw_sio_scans
- get_hw_sio_sweeps
- run_hw_sio_scan
- run_hw_sio_sweep
- remove_hw_sio_scan
- stop_hw_sio_scan
- wait_on_hw_sio_scan

Send Feedback

write_hw_sio_sweep

Write sweep data to a directory.

Syntax

```
write_hw_sio_sweep [-force] [-quiet] [-verbose] <directory> <hw_sio_sweep>
```

Returns

Name of the output directory.

Usage

Name	Description
[-force]	Overwrite existing directory
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<directory>	hardware SIO_sweep directory path
<hw_sio_sweep>	hardware SIO sweep data object

Categories

Hardware

Description

Write the populated hw_sio_sweep object after `run_hw_sio_sweep` completes.

To analyze the margin of a given link, it is often helpful to run a scan of the link using the specialized features of Xilinx UltraScale devices or 7 series FPGAs. It can also be helpful to run multiple scans on a the link with different configuration settings for the GTs. This can help you determine which settings are best for your design. The Vivado serial I/O analyzer feature enables you to define, run, and save link sweeps, or collections of link scans run across a range of values.

This command saves the specified link sweep object to disk after it has been populated by the `run_hw_sio_sweep` command.

This command returns the name of the directory created, or returns an error if the command fails.

Arguments

`-force` - (Optional) Overwrite the specified directory if it already exists.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the set_msg_config command.

<directory> - (Required) The name of a directory to write multiple scan files as a result of the run_hw_sio_sweep command.

<hw_sio_sweep> - (Required) Specify a hw_sio_sweep object to write to disk. The hw_sio_sweep must be specified as an object as returned by the get_hw_sio_sweeps command.

Example

The following example writes the specified hw_sio_sweep object to disk:

```
write_hw_sio_sweep sweep_results [get_hw_sio_sweeps {SWEEP_1}]
```

See Also

- create_hw_sio_scan
- create_hw_sio_sweep
- current_hw_device
- get_hw_sio_scans
- get_hw_sio_sweeps
- remove_hw_sio_sweep
- run_hw_sio_sweep
- stop_hw_sio_sweep
- wait_on_hw_sio_sweep

write_hw_svf

Generate SVF file for current_hw_target

Syntax

```
write_hw_svf [-force] [-quiet] [-verbose] <file_name>
```

Usage

Name	Description
[-force]	overwrite svf file if it exists
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<file_name>	SVF filename

Categories

Hardware

Description

The Vivado hardware manager supports programming of hardware devices through the use of Serial Vector Format (SVF) files. SVF files are ASCII files that contain both programming instructions and configuration data. These files are used by ATE machines and embedded controllers to perform boundary-scan operations. The SVF file captures the JTAG commands needed to program the bitstream directly into a Xilinx device, or indirectly into a flash memory device. The SVF file can be written using the `write_hw_svf` command, or applied to an open hw_target through the `execute_hw_svf` command. Refer to the *Vivado Design Suite User Guide: Programming and Debugging* (UG908) for more information.

The specific process for creating the hw_svf file is:

1. Create an SVF target using `create_hw_target`.

2. Open the SVF target.

3. Create one or more devices on the SVF target using `create_hw_device`.

4. Program the devices using commands like `program_hw_devices`.

5. Write the SVF file of operation commands using `write_hw_svf`.

In programming the hw_devices in Step 4 above, the SVF commands for the operations are cached to a temporary file. The `write_hw_svf` command saves the cache by giving it a file name and moving it to the specified file path.

Send Feedback

Note: Because this command is essentially flushing the cached SVF commands, after you use the `write_hw_svf` command, the cache is cleared, and restarted to capture any new device commands.

This command returns a message indicating success, or returns an error if it fails.

Arguments

`-force` - (Optional) Overwrite an existing file of the same name.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file_name>` - Specifies the SVF file name to write. You should specify a suffix (`.svf`) for the file as one will not be automatically assigned.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

Examples

The following example writes an SVF file to specified location:

```
program_hw_devices [lindex [get_hw_devices] 0]
write_hw_svf C:/Data/k7_design.svf
```

This example demonstrates the correct order of creating multiple devices on an SVF target. An SVF target is created and opened, then a Xilinx device, a user part, and a second Xilinx device are created on the current target. The bitstream properties are defined for the two Xilinx devices, the devices are programmed, and an SVF file is written:

```
open_hw
connect_hw_server
create_hw_target my_svf_target
open_hw_target
create_hw_device -part xc7k325t
create_hw_device -idcode 01234567 -irlength 8 -mask ffffffff -part userPart1
create_hw_device -part xcku9p
set_property PROGRAM.FILE {C:/Data/k7_design.bit} [lindex [get_hw_devices] 0]
set_property PROGRAM.FILE {C:/Data/ku_design.bit} [lindex [get_hw_devices] 2]
program_hw_devices [lindex [get_hw_devices] 0]
program_hw_devices [lindex [get_hw_devices] 2]
write_hw_svf C:/Data/myDesign.svf
```

The following example demonstrates creating a device on an SVF target, creating a configuration memory object (hw_cfgmem) associated with that device, programming the device and configuration memory, and saving that command sequence to an SVF file:

```
create_hw_target my_svf_target
open_hw_target
set device [create_hw_device -part xc7k325t]
set_property PROGRAM.FILE {C:/Data/k7_design.bit} $device
create_hw_cfgmem -hw_device $device -mem_dev  [lindex \
[get_cfgmem_parts {28f00am29ew-bpi-x16}] 0]
set cfgMem [current_hw_cfgmem]
set_property PROGRAM.ADDRESS_RANGE  {use_file} $cfgMem
set_property PROGRAM.BLANK_CHECK  0 $cfgMem
set_property PROGRAM.BPI_RS_PINS {none} $cfgMem
set_property PROGRAM.CFG_PROGRAM  1 $cfgMem
set_property PROGRAM.CHECKSUM  0 $cfgMem
set_property PROGRAM.ERASE  1 $cfgMem
set_property PROGRAM.UNUSED_PIN_TERMINATION {pull-none} $cfgMem
set_property PROGRAM.VERIFY  1 $cfgMem
set_property PROGRAM.FILES [list {C:/data/flash.mcs} ] $cfgMem
create_hw_bitstream -hw_device $device [get_property \
PROGRAM.HW_CFGMEM_BITFILE $device]
program_hw_devices $device
program_hw_cfgmem -hw_cfgmem $cfgMem
write_hw_svf C:/Data/myDesign.svf
```

See Also

- create_hw_bitstream
- create_hw_cfgmem
- create_hw_device
- create_hw_target
- execute_hw_svf
- get_cfgmem_parts
- open_hw_target
- program_hw_cfgmem
- program_hw_devices
- set_property

Send Feedback

write_hwdef

Writes hardware definition for use in the software development

Syntax

```
write_hwdef [-force] [-quiet] [-verbose] <file>
```

Returns

Success/failure status of applied action.

Usage

Name	Description
[-force]	Overwrites the existing hardware definition file
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<file>	Hardware definition file (Values: A filename with alphanumeric characters and .hwdef extention.)

Categories

Project

Description

Writes a hardware definition (`.hwdef`) file for use in the software development tools (SDK).

Each BD and HIP generates a `.hwdef` file. `Write_hwdef` merges all the files into a single container. This command is run automatically by the Vivado Design Suite when generating the output products for a top-level design that includes a block design with an embedded processor like MicroBlaze, or Zynq-7000 SoC. Block designs are created in the IP integrator of the Vivado Design Suite with the `create_bd_design` command.

The `write_hwdef` command is run after `place_design` and creates a hardware container file with `.hwdef` extension. The container file includes device metadata and hardware design files.

The `write_hwdef` command returns nothing if successful, or an error if the command fails.

Arguments

`-force` - (Optional) Overwrite and existing hardware definition file if one exists. If this option is not specified, then the Vivado Design Suite will not overwrite an existing file.

Send Feedback

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file>` - (Required) Specify the name of the hardware definition file. The file can include the path and file extension. The default file extension of `.hwdef` is used if an extension is not specified.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

Examples

The following example creates the specified hardware definition file:

```
write_hwdef  -force C:/Data/ug940/lab1/zynq_design.hdf
```

See Also

- create_bd_design

write_ibis

Write IBIS models for current floorplan

Syntax

```
write_ibis [-force] [-allmodels] [-nopin] [-no_pin_mapping]
    [-truncate <arg>] [-component_name <arg>] [-ibs <arg>] [-pkg <arg>]
    [-quiet] [-verbose] <file>
```

Returns

Name of the output file.

Usage

Name	Description
[-force]	Overwrite existing .ibs file
[-allmodels]	Include all available buffer models for this architecture. By default, only buffer models used by the floorplan are included.
[-nopin]	Disable inclusion of the per-pin modeling of the package (path from the die pad to the package pin). Package is reduced to a single RLC transmission line model applied to all pins and defined in the [Package] section. Default: This option is not set. IBISWriter includes per-pin modeling of the package as RLC matrices in the [Define Package Model] section if this data is available.
[-no_pin_mapping]	Do not output the [Pin Mapping] section for ultrascale, ultrascale plus, and versal.
[-truncate]	Maximum length for a signal name in the output file. Names longer than this will be truncated. This property can be set to truncate signal name length to 20, 40, or 0 (unlimited). Default: Truncate signal name length to 40 characters in accordance with the IBIS version 4.2 specification. Default: 40
[-component_name]	Specify a new component name for use in multiple FPGA designs to replace the default.
[-ibs]	Specify an updated generic IBIS models file.
[-pkg]	Specify an updated per pin parasitic package data file.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<file>	Output file name. The .ibs extension is optional.

Categories

[FileIO](#)

Send Feedback

Description

Writes the IBIS models for the target device in the current design. The netlist and implementation details from the design are combined with the per-pin parasitic package information to create a custom IBIS model for the design.

Because the `write_ibis` command incorporates design information into the IBIS Model, you must have an RTL, Netlist, or Implemented Design open when running this command.

Arguments

`-force` - (Optional) Overwrite the IBIS file if it already exists.

`-allmodels` - (Optional) Export all buffer models for the target device. By default the tool will only write buffer models used by the design.

`-nopin` - (Optional) Disable per-pin modeling of the path from the die pad to the package pin. The IBIS model will include a single RLC transmission line model representation for all pins in the [Package] section. By default the file will include per-pin modeling of the package as RLC matrices in the [Define Package Model] section if this data is available.

`-no_pin_mapping` - (Optional) Do not output the Pin Mapping section for UltraScale, UltraScale+, and Versal devices.

`-truncate <arg>` - (Optional) The maximum length for a signal name in the output file. Names longer than this will be truncated. Valid values are 20, 40, or 0 (unlimited). By default the signal names are truncated to 40 characters in accordance with the IBIS version 4.2 specification.

`-component_name <arg>` - (Optional) Specify a new component name to change the default value, which is the device family.

`-ibs <arg>` - (Optional) Specify an updated generic IBIS models file. This is used to override the IBIS models found in the tool installation under the parts directory. This argument is required for any parts that do not have generic models in the installation directory.

`-pkg <arg>` - (Optional) Specify an updated per pin parasitic package data file. This is used to override the parasitic package file found in the tool installation hierarchy under the parts directory. This argument is required for any parts that do not have generic models in the installation directory.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file>` - (Required) The filename of the IBIS file to write.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

Examples

The following example exports all buffer models for the target device, eliminates truncation of signal names, and specifies the file name and path to write:

```
write_ibis -allmodels -truncate 0 C:/Data/FPGA_Design/ibisOut.txt
```

write_inferred_xdc

Write file with inferred xdc timing constraints

Syntax

```
write_inferred_xdc [-force] [-all] [-append] [-async_clocks]
    [-all_async_reg] [-clock_groups] [-clocks] [-excl_clocks] [-exceptions]
    [-io_constraints] [-merge_existing_constraints] [-name <arg>] [-quiet]
    [-verbose] [<file>]
```

Usage

Name	Description
[-force]	Overwrite existing file.
[-all]	Generate all constraints except missing clocks which are generated with the -clocks option
[-append]	Append the constraints to file, don't overwrite the constraints file
[-async_clocks]	Find asynchronous clock groups
[-all_async_reg]	Find the missing ASYNC_REG property for safe and unsafe Clock Domain Crossing
[-clock_groups]	Find asynchronous and exclusive clock groups, equivalent to options -async_clocks -excl_clocks
[-clocks]	Find missing clock definitions
[-excl_clocks]	Find logically and physically exclusive clock groups
[-exceptions]	Find missing exceptions
[-io_constraints]	Find missing input and output delays
[-merge_existing_constraints]	Add existing user defined constraints to the generated constraints
[-name]	Start constraints wizard in a GUI panel with this name. Do other command options can be combined with -name.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<file>]	Filename to write constraints into

Categories

[FileIO](), [Timing]()

Description

You can use the `write_inferred_xdc` to find constraints that should be defined in the open synthesized or implemented design. Write timing constraints that are automatically generated by the Vivado timing engine, rather than defined in an existing XDC file and added to the design.

Run `write_inferred_xdc -clocks` first to define suggested clock and generated clock constraints. The suggested clock constraints will be defined with a period of 1 ns. You can edit the recommended constraints to create clocks and generated clocks with the required clock period to meet the needs of your design.

You can add the edited constraints file into the design using `read_xdc`, or `add_files`, and `update_timing`.

You may need to run the `write_inferred_xdc` command multiple times, using various options like `-clock_groups` or `-async_clocks`, to capture all inferred timing constraints from the fully clocked design. You can use an iterative process of writing and sourcing the inferred clocked constraints, and then writing and sourcing additional constraint files to capture all inferred constraints. See the example below for more information.

This command returns a transcript of the process when successful, or returns an error if it fails.

Arguments

`-force` - (Optional) Overwrite the specified output file if it already exists.

`-all` - (Optional) Write all XDC constraints for the current design, except missing clocks. The missing clocks can be separately obtained using the `-clocks` option.

`-append` - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note: The `-append` option can only be used with the `-file` option.

`-async_clocks` - (Optional) Find asynchronous clock groups that should be defined using the `set_clock_groups -asynchronous` constraint.

`-all_async_reg` - (Optional) Find missing ASYNC_REG properties for safe and unsafe Clock Domain Crossing.

`-clock_groups` - (Optional) Find asynchronous and exclusive clock groups, equivalent to specifying both the `-async_clocks` and `-excl_clocks` options in the same command.

`-clocks` - (Optional) Find missing clock and generated clock definitions that should be defined by the `create_clock` and `create_generated_clock` constraints.

> 💡 **TIP:** *This is the default report generated by the* `write_inferred_xdc` *command when no other constraint options are specified.*

`-excl_clocks` - (Optional) Find physically and logically exclusive clock groups that should be defined using the `set_clock_groups -physically_exclusive` constraint or the `set_clock_groups -logically_exclusive` constraint.

`-exceptions` - (Optional) Find missing timing exceptions that should be defined by timing constraints such as `set_false_path` or `set_multicycle_path` that change the default assumptions for timing paths in the design.

`-io_constraints <arg>` - (Optional) Find missing I/O constraints such as `set_input_delay`, and `set_output_delay`.

`-merge_existing_constraints` - (Optional) This option reports existing user-defined constraints matching the type of inferred constraints currently being reported by the `write_inferred_xdc` command.

> **TIP:** *The existing constraints are written to the specified file as comments so that the inferred constraints file can be read into the Vivado Design Suite using `read_xdc` without conflicting with existing constraints.*

`-name <arg>` - (Optional) Specifies the name of the results set to return the reported constraints to when running in the Vivado IDE.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file>` - (Required) The filename to write the inferred XDC constraints to. You should specify a file extension as part of the file name, as the `write_inferred_xdc` command will not provide one.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

Examples

The following example writes the inferred clock constraints in the current design:

```
write_inferred_xdc -clocks C:/Data/design1_inferred_clocks.xdc
```

The `write_inferred_xdc` command may need to be run multiple times to capture all the inferred constraints, as is shown in this example:

```
write_inferred_xdc -clocks clocks.xdc
source clocks.xdc
write_inferred_xdc -all all.xdc
source all.xdc
write_inferred_xdc -async_clocks async.xdc
source async.xdc
```

See Also

- create_clock
- create_generated_clock
- report_exceptions
- report_timing
- set_clock_groups
- set_false_path
- set_input_delay
- set_multicycle_path
- set_output_delay
- write_xdc

write_ip_tcl

Write a tcl script on disk that will recreate a given IP.

Syntax

```
write_ip_tcl [-force] [-no_ip_version] [-ip_name <arg>] [-show_defaults]
    [-multiple_files] [-quiet] [-verbose] [<objects>] [<tcl_filename>...]
```

Returns

IP TCL file.

Usage

Name	Description
[-force]	Flag to overwrite existing file.
[-no_ip_version]	Flag to not include the IP version in the IP VLNV in create_ip commands. NOTE - this may have implications if there are major IP version changes.
[-ip_name]	Set the name of the IP. This argument is not supported for multiple IP.
[-show_defaults]	Add a comment containing the default parameter values of the IP.
[-multiple_files]	Flag to create a .tcl file for each IP supplied as an argument
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<objects>]	IP(s) to be written to disk Values: IP instance(s) as returned by 'get_ips <instance name>'
[<tcl_filename>]	File path to the exported tcl file. If the path is a directory and multiple IP are given as an argument, a file for each IP will be created. Default: ./

Categories

[Object](), [Project](), [IPFlow]()

Send Feedback

write_iphys_opt_tcl

Write iPhysOpt script.

Syntax

```
write_iphys_opt_tcl [-place] [-binary] [-quiet] [-verbose] [<output>]
```

Usage

Name	Description
[-place]	write out placement information
[-binary]	write out in binary format
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<output>]	tcl file containing iPhysOpt script

Categories

Tools

Description

Because physical optimization requires timing data that is only available after placement, the `phys_opt_design` command cannot be run prior to placement. However, the interactive physical optimization feature, or `iphys_opt_design`, lets you write out the physical optimizations performed on the post-placed design, and then apply those optimizations to the design netlist prior to placement. Refer to the *Vivado Design Suite User Guide: Implementation (UG904)* for more information on interactive physical optimization.

Interactive physical optimization can be used in two ways:

- Applying post-placement physical optimizations to the pre-placement netlist to improve the overall placement result and improve design performance.

- Saving the physical optimizations in a Tcl script to be repeated as needed.

The `write_iphys_opt_tcl` command can only be run after placement, on a design that has had actual physical optimizations performed.

> **TIP:** *You can use the* `report_phys_opt` *command to report the physical optimizations that have been performed on the design.*

The output is a Tcl script file with a sequence of `iphys_opt_design` commands listing the specific optimizations performed by the `phys_opt_design` command. The `iphys_opt` Tcl script can be edited to change the specific optimizations performed. The Tcl script provides a history of the physical optimizations performed on the design after placement, marked by date and history.

> **IMPORTANT!** *The* `iphys_opt` *Tcl script contains the specific optimizations performed by the* `phys_opt_design` *command, but does not include placement and routing changes or results.*

This command returns nothing if successful, or returns an error if it fails.

Arguments

`-place` - (Optional) Write out placement data for optimized cells in the design, as well as the physical optimization Tcl commands. The default iphys_opt Tcl script does not include the placement data.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<output> - (Required) The name of the interactive physical optimization Tcl file to write. You should specify the path, name, and extension for the file.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

Examples

The following example writes the physical optimizations that have been performed in the current design to the specified Tcl script:

```
write_iphys_opt_tcl C:/Data/myDesign_physopt.tcl
```

See Also

- iphys_opt_design
- phys_opt_design
- read_iphys_opt_tcl
- report_phys_opt

write_mem_info

Write the Memory Map Info of the design to a .mmi file.

Syntax

```
write_mem_info [-cell <arg>] [-no_partial_mmi] [-force_detect_xpm] [-force]
    [-quiet] [-verbose] <file>
```

Returns

The name of the .mmi file.

Usage

Name	Description
[-cell]	Create only partial .mmi for named cell
[-no_partial_mmi]	Do not write partial bit files for a Dynamic Function eXchange design
[-force_detect_xpm]	Traverse netlist and check for XPM
[-force]	Overwrite existing mem info xml file
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<file>	Design mem info file Values: A filename with alphanumeric characters and .mmi extension.

Categories

FileIO

Description

This command writes a memory information (MMI) file defining the BRAM placement and address ranges to create a memory map of the design.

> ⭐ **IMPORTANT!** `write_mem_info` *requires an open implemented design so that the memory information includes the BRAM placement data, as well as the address ranges, required for proper programming.*

The memory map information (MMI) file, written by the `write_mem_info` command, is a text file that describes how individual Block RAMs on the Xilinx device are grouped together to form a contiguous address space called an Address Block.

Send Feedback

The mem info file (MMI) contains memory mapping information similar to the Block Memory Map (BMM) file, but in a format that can be read by the `updatemem` command to merge with a bitstream (BIT) file. The `updatemem` command uses the MMI file to identify the physical BRAM resource that maps to a specific address range. Refer to the *Vivado Design Suite User Guide: Embedded Processor Hardware Design (UG898)* for more information on running `updatemem`.

This command returns the name of the file created, or returns an error if it fails.

Arguments

`-force` - (Optional) Overwrite an existing memory info file if one exists. If this option is not specified, then the Vivado Design Suite will not overwrite an existing file.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file>` - (Required) Specify the name of the memory info file to write. The file can include the path and file extension. The default file extension of `.mmi` is used if an extension is not specified.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

Example

The following example creates the memory mapping file from the open design:

```
write_mem_info C:/Data/design1.mmi
```

See Also

- write_bmm

write_peripheral

Save peripheral component to the disk.

Syntax

```
write_peripheral [-quiet] [-verbose] <peripheral>
```

Usage

Name	Description
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<peripheral>`	Peripheral object

Categories

[Project](#), [IPFlow](#), [CreatePeripheral](#)

Description

Write the specified AXI peripheral object to disk in the form of the `component.xml` file. The peripheral is written to the repository location specified by the `create_peripheral` command, under the name specified at creation.

Arguments

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

<peripheral> - (Required) The peripheral object to write. The peripheral is created with the `create_peripheral` command, and should be captured in a Tcl variable to facilitate further processing by this and other related commands. See the example below.

Send Feedback

Example

This example creates a new AXI peripheral, with the VLNV attribute as specified, and captures the peripheral object in a Tcl variable for later processing, then adds AXI slave interfaces to the peripheral, and generates the output products for the peripheral, and writes the component.xml file to disk. The directory of the new peripheral is added to the IP_REPO_PATH property of the current fileset, and the IP catalog is updated to include the new peripheral:

```
set perifObj [ create_peripheral {myCompany.com} {user} {testAXI1} \
    {1.3} -dir {C:/Data/new_periph} ]
add_peripheral_interface {S0_AXI} -interface_mode {slave} \
    -axi_type {lite} $perifObj
add_peripheral_interface {S1_AXI} -interface_mode {slave} \
    -axi_type {lite} $perifObj
generate_peripheral -driver -bfm_example_design \
    -enable_interrupt $perifObj
write_peripheral $perifObj
set_property ip_repo_paths  C:/Data/new_periph [current_fileset]
update_ip_catalog -rebuild
```

See Also

- add_peripheral_interface
- create_peripheral
- generate_peripheral

write_project_tcl

(User-written application) Export Tcl script for re-creating the current project

Syntax

```
write_project_tcl [-paths_relative_to <arg>] [-origin_dir_override <arg>]
    [-target_proj_dir <arg>] [-force] [-all_properties] [-no_copy_sources]
    [-no_ip_version] [-absolute_path] [-dump_project_info] [-use_bd_files]
    [-internal] [-validate] [-quiet] [-verbose] <file>
```

Returns

True (0) if success, false (1) otherwise.

Usage

Name	Description
[-paths_relative_to]	Override the reference directory variable for source file relative paths Default: Script output directory path
[-origin_dir_override]	Set 'origin_dir' directory variable to the specified value (Default is value specified with the -paths_relative_to switch) Default: None
[-target_proj_dir]	Directory where the project needs to be restored Default: Current project directory path
[-force]	Overwrite existing tcl script file
[-all_properties]	Write all properties (default & non-default) for the project object(s)
[-no_copy_sources]	Do not import sources even if they were local in the original project Default: 1
[-no_ip_version]	Flag to not include the IP version as part of the IP VLNV in create_bd_cell commands. Default: 1
[-absolute_path]	Make all file paths absolute wrt the original project directory
[-dump_project_info]	Write object values
[-use_bd_files]	Use BD sources directly instead of writing out procs to create them
[-internal]	Print basic header information in the generated tcl script
[-validate]	Runs a validate script before recreating the project. To test if the files and paths refrenced in the tcl file exists or not.
[-quiet]	Execute the command quietly, returning no messages from the command.
[-verbose]	Suspend message limits during command execution
<file>	Name of the tcl script file to generate

Categories

[xilinxtclstore](#), [projutils](#)

Send Feedback

Description

Creates a Tcl script to recreate the current project.

The generated script will contain the Tcl commands for creating the project, setting the project type, creating filesets, adding/importing source files, defining runs and run properties.

> ⭐ **IMPORTANT!** *The new project will be created in the current working directory (CWD) where the generated Tcl script is sourced from. The script written out by `write_project_tcl` should be sourced in the same directory from which it was created. If you source the script from a different directory, you should first `set` the <origin_dir_loc> variable in Tcl shell to this alternate directory, or edit the script to define the <origin_dir> variable in the script in order to maintain the relative path between the CWD and the source files referenced in the script.*

This Tcl project script and the various design sources can be stored in a version control system for source file management and project archival.

Arguments

`-paths_relative_to <arg>` - (Optional) Change the 'origin_dir' variable for the relative paths of source files.

`-origin_dir_override <arg>` - (Optional) Override the 'origin_dir' directory variable to the specified value. The default is the value specified with the `-paths_relative_to` option.

`-target_proj_dir <arg>` - (Optional) Specify the directory path where the project will be recreated. The tool will write the `create_project` command using the directory path specified with this option. The default is the current project directory.

`-force` - (Optional) Overwrite an existing project script file of the same name. If the script file already exists, the tool returns an error unless the `-force` argument is specified.

`-all_properties` - (Optional) Write all properties (default and non-default) for the project. The tool will write `set_property` commands for all the objects like project, filesets, files, runs etc.

> 💡 **TIP:** *By default, if the `-all_properties` switch is not specified, only non-default properties will be written to the script.*

`-no_copy_sources` - (Optional) Do not import sources even if they are local to the original project. The tool will not import the files that were local in the original project into the new project.

> ⭐ **IMPORTANT!** *This option requires the addition of the `-use_bd_files` switch for designs that contain block diagrams.*

`-absolute_path` - (Optional) Make all file paths absolute with regard to the original project directory.

`-dump_project_info` - (Optional) Writes a file containing a dump of the current values of all objects in the design, and well as a file indicating what the default values are for all objects in the design. These files can be used to aide in debugging issues with the recreated project.

`-use_bd_files` - (Optional) Use block diagram sources (`.bd`) from the Vivado IP integrator directly instead of writing out Tcl scripts to recreate them. By default the `write_project_tcl` command will recreate the block diagram as a Tcl proc inside the project Tcl script. With the use of the `-use_bd_files` option, the block design sources will be copied or referenced like all other design sources according to the `-no_copy_sources` option.

`-internal` - (Optional) Includes basic header information in the generated Tcl script.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file>` - (Required) The name of the script file to be created by the `write_project_tcl` command. The tool will apply an extension of `.tcl` if a file extension is not supplied.

Examples

The following example exports Tcl script named `recreate.tcl` for the current project:

```
write_project_tcl recreate.tcl
```

The following example exports a Tcl script named `recreate.tcl` for the current project in the `./script` directory and specifies the `/tmp/test` directory for the `create_project` command. When the `recreate.tcl` script is run in the Vivado Tcl shell, the project will be re-created in `/tmp/test` directory:

```
write_project_tcl -target_proj_dir "/tmp/test" ./script/recreate.tcl
```

The following command exports Tcl script for the current project and writes all the properties, both default or non-default values:

```
write_project_tcl -all_properties recreate.tcl
```

The following command exports Tcl script for the current project and adds files that are local in this project. The recreated project will reference these files:

```
write_project_tcl -no_copy_sources -use_bd_files recreate.tcl
```

> **IMPORTANT!** *The* `-use_bd_files` *switch is required for use with* `-no_copy_sources` *in designs with block diagrams.*

The following command exports `recreate.tcl` script for the current project in the current working directory, creates a new project in `./my_test` directory, prints the list of files in the new project, prints the current project settings and then closes the newly created project:

```
open_project ./test/test.xpr
write_project_tcl -force recreate.tcl
close_project
file mkdir my_test
cd my_test
source ../recreate.tcl
get_files -of_objects [get_filesets sources_1]
report_property [current_project]
close_project
```

The following command creates a new project named bft_test, adds files to the project, sets the fileset property, exports a tcl script named `bft.tcl` in the current working directory, creates a new project in `./my_bft` directory, prints the list of files in the new project (`test_1.v` and `test_2.v`), prints the "verilog_define" property value and then closes the newly created project:

```
create_project bft_test ./bft_test
add_files test_1.v
add_files test_2.v
set_property verilog_define {a=10} [get_filesets sources_1]
write_project_tcl -force bft.tcl
close_project
file mkdir my_bft
cd my_bft
source ../bft.tcl
get_files -of_objects [get_filesets sources_1]
get_property verilog_define [get_filesets sources_1]
close_project
```

See Also

- add_files
- archive_project
- close_project
- create_project
- current_project
- get_files
- get_property
- open_project
- report_property
- set_property

Send Feedback

write_qor_suggestions

Write QoR Suggestions to the given file

Syntax

```
write_qor_suggestions [-strategy_dir <arg>] [-tcl_output_dir <arg>]
    [-force] [-of_objects <args>] [-disable_dont_touch] [-quiet] [-verbose]
    <file>
```

Usage

Name	Description
[-strategy_dir]	Directory to create Strategy RQS & TCL files Values: If passed a directory path, for each strategy suggested one set of RQS and TCL files will be generated.
[-tcl_output_dir]	Directory to create TCL files Values: TCL files for the QoR suggestions will be generated in the provided directory.
[-force]	Overwrite existing suggestions file
[-of_objects]	List of QoR suggestion objects
[-disable_dont_touch]	write suggestion command having DONT_TOUCH to file
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<file>	QoR suggestions file Values: A filename with alphanumeric characters and .rqs extension.

Categories

[FileIO](), [Feasibility](), [Timing]()

Description

Write the QoR suggestions generated by the `report_qor_suggestions` command. You can combine the suggestions from the latest report with suggestions read into the design with `read_qor_suggestions` so that you can manage all suggestions in a single RQS file.

To write out specific QoR suggestions, use the `-of_objects` option. When this is not specified, all suggestions will be written.

The recommended way to manage suggestions is using RQS objects. However, it is possible to view and execute the commands using Tcl. Specifying the `-tcl_output_dir` option writes Tcl scripts for the automated suggestions that are property based.

Send Feedback

Implementation strategies that use machine learning to analyze the design can be generated running `report_qor_suggestions`. If you specify the `-strategy_dir` option, multiple Tcl files and one RQS file will be written for each strategy. The Tcl files aid integration into project or non project flows. The main RQS file should not be used as the suggestions are contained in the run specific files along with strategy information.

This command returns the name of the output file created when successful, or returns an error if it fails.

Arguments

`-strategy_dir <arg>` - (Optional) Creates Tcl scripts and RQS files containing commands to set up implementation runs based on machine learning.

`-tcl_output_dir <arg>` - (Optional) Creates Tcl scripts containing XDC commands.

`-force` - (Optional) Overwrite the specified file if it already exists.

`-of_objects <args>` - (Optional) Writes the specified QoR suggestion objects as returned by the `get_qor_suggestions` command. This can be used to write specific QoR suggestions based on specific properties.

`-disable_dont_touch` - (Optional) Allows suggestions to be generated that reset the `DONT_TOUCH` property. This may lead to slightly lower MTBF and signals no longer being available.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file>` - (Optional) Write the QoR suggestions to the specified file. If no file extension is specified, the `.rqs` extension will be added. If the specified file exists, an error will be returned unless the `-force` option is also specified.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

Note: At least one of `strategy_dir`, `tcl_output_dir`, or a file must be specified.

Examples

Report QoR suggestions and write non-strategy suggestions to the specified file.

```
report_qor_suggestions
write_qor_suggestions C:/Data/qor_results.rqs
```

Report QoR suggestions and write both strategy and non-strategy suggestions.

```
report_qor_suggestions
write_qor_suggestions -strategy_dir C:/Data/strategy_dir C:/Data/
qor_suggestions.rqs
```

This creates one RQS file for each strategy suggestion, generating up to three by default. Each RQS file includes one strategy suggestion and all non-strategy suggestions. This means that for subsequent runs, one RQS file per run is required.

Note: To make use of strategy suggestions, the directive for each implementation command (`opt_design`, `place_design`, `phys_opt_design`, and `route_design`) must be set to RQS. This can be configured automatically in project mode by sourcing the project mode Tcl script generated for each RQS file in the `strategy_dir`. There is also a non-project example Tcl script that demonstrates this requirement.

See Also

- get_qor_suggestions
- read_qor_suggestions
- report_qor_suggestions

Send Feedback

write_schematic

Export schematic

Syntax

```
write_schematic [-force] [-format <arg>] [-orientation <arg>]
    [-scope <arg>] [-name <arg>] [-quiet] [-verbose] <file>
```

Returns

Name of the output file.

Usage

Name	Description
[-force]	Overwrite existing file
[-format]	Values: native or pdf. read_schematic can be used to view native format. Default: native
[-orientation]	Values: landscape or portrait
[-scope]	Values: current_page, visible or all Default: current_page
[-name]	Schematic window title
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<file>	Output file

Categories

[FileIO](#)

Description

Output the currently opened, or specified Schematic window in the Vivado IDE to a file.

The file can be written as a native ASCII file that can be read back into the Vivado IDE using the read_schematic command, or can be written as a PDF or SVG file for use outside of the Vivado Design Suite. This can be useful when documenting IP cores from the IP packager flow, or from the Vivado IP integrator.

Arguments

`-force` - (Optional) Overwrite the specified Schematic file if it already exists.

`-format [ native | pdf | svg ]` - (Optional) Write the output file in the specified format. The default file format is `native` to the Vivado Design Suite, that can be read back into the tool using the `read_schematic` command. `SVG` is scalable vector graphics format. `PDF` is portable document format.

Note: The `-format` argument is case sensitive.

`-orientation [ landscape | portrait ]` - (Optional) Write the schematic in a vertical (portrait) or horizontal (landscape) orientation. The default is `portrait`.

`-scope [ current_page | visible | all ]` - (Optional) Write the current page of the schematic, or the visible content displayed in the schematic window, or all of the schematic. The default is `current_page`.

`-name <arg>` - (Optional) Specifies the name of the open Schematic window to write. Use this option to write the specified schematic window when more than one schematic window is opened.

Note: When multiple schematic windows are opened, the first opened window is written if `-name` is not specified.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file>` - (Required) The path and filename of the Schematic file to write. The path is optional, but if one is not provided the Schematic file will be written to the current working directory, or the directory from which the Vivado tool was launched.

Example

The following example writes the specified Schematic window, displayed in the Vivado IDE, to a native format file that can later be read back into the Vivado Design Suite, overwriting the specified file if it already exists:

```
write_schematic -name Schematic C:/Data/mySchematic.txt -force
```

The following example writes a PDF file with a horizontal view of the specified schematic window:

```
write_schematic -format pdf -name "Schematic (2)" C:/Data/mySchematic.pdf \
-orientation landscape
```

See Also

- read_schematic
- write_bd_layout

write_sdf

write_sdf command generates flat sdf delay files for event simulation

Syntax

```
write_sdf [-process_corner <arg>] [-cell <arg>] [-rename_top <arg>]
    [-force] [-mode <arg>] [-gzip] [-multithread <arg>] [-quiet] [-verbose]
    <file>
```

Usage

Name	Description
[-process_corner]	Specify process corner for which SDF delays are required; Values: slow, fast Default: slow
[-cell]	Root of the design to write, e.g. des.subblk.cpu Default: whole design
[-rename_top]	Replace name of top module with custom name e.g. netlist Default: new top module name
[-force]	Overwrite existing SDF file
[-mode]	Specify sta (Static Timing Analysis) or timesim (Timing Simulation) mode for SDF Default: timesim
[-gzip]	write gzipped SDF
[-multithread]	Run in multithread mode with specified number of threads Default: -1
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<file>	File name

Categories

FileIO, Simulation, Timing

Description

Writes the timing delays for cells in the design to a Standard Delay Format (SDF) file.

The output SDF file can be used by the `write_verilog` command to create Verilog netlists for static timing analysis and timing simulation.

Arguments

`-process_corner` [fast | slow] - (Optional) Write delays for a specified process corner. Delays are greater in the slow process corner than in the fast process corner. Valid values are 'slow' or 'fast'. By default, the SDF file is written for the slow process corner.

`-cell <arg>` - (Optional) Write the SDF file from a specific cell of the design hierarchy. The default is to create an SDF file for the whole design.

`-rename_top <arg>` - (Optional) Rename the top module in the output SDF file as specified.

`-force` - (Optional) Forces the overwrite of an existing SDF file of the same name.

`-mode` [timesim | sta]- (Optional) Specifies the mode to use when writing the SDF file. Valid values are:

- timesim - Output an SDF file to be used for timing simulation. This is the default setting.

- sta - Output an SDF file to be used for static timing analysis (STA).

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file>` - (Required) The file name of the SDF file to write. The SDF file is referenced in the Verilog netlist by the use of the `-sdf_anno` and `-sdf_file` arguments of the `write_verilog` command.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

Examples

The following example writes an SDF file to the specified directory:

```
write_sdf C:/Data/FPGA_Design/designOut.sdf
```

See Also

- write_verilog

Send Feedback

write_verilog

Export the current netlist in Verilog format

Syntax

```
write_verilog [-cell <arg>] [-mode <arg>] [-lib] [-port_diff_buffers]
    [-write_all_overrides] [-keep_vcc_gnd] [-rename_top <arg>]
    [-sdf_anno <arg>] [-sdf_file <arg>] [-force] [-include_xilinx_libs]
    [-logic_function_stripped] [-quiet] [-verbose] <file>
```

Returns

The name of the output file or directory.

Usage

Name	Description
[-cell]	Root of the design to write, e.g. des.subblk.cpu Default: whole design
[-mode]	Values: design, pin_planning, synth_stub, sta, funcsim, timesim Default: design
[-lib]	Write each library into a separate file
[-port_diff_buffers]	Output differential buffers when writing in -port mode
[-write_all_overrides]	Write parameter overrides on Xilinx primitives even if the override value is the same as the default value
[-keep_vcc_gnd]	Don't replace VCC/GND instances by literal constants on load terminals. For simulation modes only.
[-rename_top]	Replace top module name with custom name e.g. netlist Default: new top module name
[-sdf_anno]	Specify if sdf_annotate system task statement is generated
[-sdf_file]	Full path to sdf file location Default: <file>.sdf
[-force]	Overwrite existing file
[-include_xilinx_libs]	Include simulation models directly in netlist instead of linking to library
[-logic_function_stripped]	Convert INIT strings on LUTs & RAMBs to fixed values. Resulting netlist will not behave correctly.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<file>	Which file to write

Categories

[FileIO](#), [Simulation](#)

Description

Write a Verilog netlist of the current design or from a specific cell of the design to the specified file or directory. The output is a IEEE 1364-2001 compliant Verilog HDL file that contains netlist information obtained from the input design files.

You can output a complete netlist of the design or specific cell, or output a port list for the design, or a Verilog netlist for simulation or static timing analysis.

Arguments

`-cell <arg>` - (Optional) Write the Verilog netlist from a specified cell or block level of the design hierarchy. The output Verilog file or files will only include information contained within the specified cell or module.

`-mode <arg>` - (Optional) The mode to use when writing the Verilog file. By default, the Verilog netlist is written for the whole design. Valid mode values are:

- design - Output a Verilog netlist for the whole design. This acts as a snapshot of the design, including all post placement, implementation, and routing information in the netlist.

- pin_planning - Output only the I/O ports for the top-level of the design.

- synth_stub - Output the ports from the top-level of the design for use as a synthesis stub.

- sta - Output a Verilog netlist to be used for static timing analysis (STA).

- funcsim - Output a Verilog netlist to be used for functional simulation. The output netlist is not suitable for synthesis.

- timesim - Output a Verilog netlist to be used for timing simulation. The output netlist is not suitable for synthesis.

`-lib` - (Optional) Create a separate Verilog file for each library used by the design.

Note: The `-library` option can only be used for simulation. Vivado synthesis will treat all Verilog files as being in the default work library.

`-port_diff_buffers` - (Optional) Add the differential pair buffers and internal wires associated with those buffers into the output ports list. This argument is only valid when `-mode pin_planning` or `-mode synth_stub` is specified.

`-write_all_overrides [ true | false ]` - (Optional) Write parameter overrides, in the design to the Verilog output even if the value of the parameter is the same as the defined primitive default value. If the option is false then parameter values which are equivalent to the primitive defaults are not output to the Verilog file. Setting this option to true will not change the result but makes the output Verilog more verbose.

Send Feedback

`-keep_vcc_gnd` - (Optional) By default, when writing a nelist for simulation, or from an IP integrator block design, the Vivado Design Suite replaces VCC and GND primitives, and the nets they drive, with literal constants on each of the loads on the net. The `-keep_vcc_gnd` option disables this default behavior and preserves the VCC or GND primitives.

`-rename_top <arg>` - (Optional) Rename the top module in the output as specified. This option only works with `-mode funcsim` or `-mode timesim` to allow the Verilog netlist to plug into top-level simulation test benches.

`-sdf_anno` [true | false] - (Optional) Add the `$sdf_annotate` statement to the Verilog netlist. Valid values are true (or 1) and false (or 0). This option only works with `-mode timesim`, and is set to false by default.

`-sdf_file <arg>` - (Optional) The path and filename of the SDF file to use when writing the `$sdf_annotate` statement into the output Verilog file. When not specified, the SDF file is assumed to have the same name and path as the Verilog output specified by `<file>`, with a file extension of `.sdf`. The SDF file must be separately written to the specified file path and name using the `write_sdf` command.

`-force` - (Optional) Overwrite the Verilog files if they already exists.

`-include_xilinx_libs` - (Optional) Write the simulation models directly in the output netlist file rather than pointing to the libraries by reference.

`-logic_function_stripped` - (Optional) Hides the INIT values for LUTs & RAMs by converting them to fixed values in order to create a netlist for debug purposes that will not behave properly in simulation or synthesis.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file>` - (Required) The path and filename of the Verilog file to write. The path is optional, but if one is not provided the Verilog file will be written to the current working directory, or the directory from which the Vivado tool was launched.

Examples

The following example writes a Verilog simulation netlist file for the whole design to the specified file and path:

```
write_verilog C:/Data/my_verilog.v
```

In the following example, because the `-mode timesim` and `-sdf_anno` options are specified, the `$sdf_annotate` statement will be added to the Verilog netlist. However, since the `-sdf_file` option is not specified, the SDF file is assumed to have the same name as the Verilog output file, with an `.sdf` file extension:

```
write_verilog C:/Data/my_verilog.net -mode timesim -sdf_anno true
```

Note: The SDF filename written to the `$sdf_annotate` statement will be `my_verilog.sdf`.

In the following example, the functional simulation mode is specified, the option to keep VCC and GND primitives in the output simulation netlist is enabled, and the output file is specified:

```
write_verilog -mode funcsim -keep_vcc_gnd out.v
```

See Also

- write_sdf
- write_vhdl

write_vhdl

Export the current netlist in VHDL format

Syntax

```
write_vhdl [-cell <arg>] [-mode <arg>] [-lib] [-port_diff_buffers]
    [-write_all_overrides] [-keep_vcc_gnd] [-rename_top <arg>] [-arch_only]
    [-force] [-include_xilinx_libs] [-quiet] [-verbose] <file>
```

Returns

The name of the output file or directory.

Usage

Name	Description
[-cell]	Root of the design to write, e.g. des.subblk.cpu Default: whole design
[-mode]	Output mode. Valid values: funcsim, pin_planning, synth_stub Default: funcsim
[-lib]	Write each library into a separate file
[-port_diff_buffers]	Output differential buffers when writing in -port mode
[-write_all_overrides]	Write parameter overrides on Xilinx primitives even if the same as the default value
[-keep_vcc_gnd]	Don't replace VCC/GND instances by literal constants on load terminals. For simulation modes only.
[-rename_top]	Replace top module name with custom name e.g. netlist Default: new top module name
[-arch_only]	Write only the architecture, not the entity declaration for the top cell
[-force]	Overwrite existing file
[-include_xilinx_libs]	Include simulation models directly in netlist instead of linking to library
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
<file>	Which file to write

Categories

[FileIO](), [Simulation]()

Description

Write a VHDL netlist of the current design or from a specific cell of the design to the specified file or directory.

Send Feedback

The output of this command is a VHDL IEEE 1076.4 VITAL-2000 compliant VHDL file that contains netlist information obtained from the input design files. You can output a complete netlist of the design or specific cell, or output a port list for the design.

Arguments

`-cell <arg>` - (Optional) Write the VHDL netlist from a specified cell or block level of the design hierarchy. The output VHDL file or files will only include information contained within the specified cell or module.

`-mode <arg>` - (Optional) The mode to use when writing the VHDL file. By default, the simulation netlist is written for the whole design. Valid mode values are:

- funcsim - Output the VHDL netlist to be used as a functional simulation model. The output netlist is not suitable for synthesis. This is the default setting.

- pin_planning - Output only the I/O ports in the entity declaration for the top module.

- synth_stub - Output the ports from the top-level of the design for use as a synthesis stub.

`-lib` - (Optional) Create a separate VHDL file for each library used by the design.

Note: This option is the opposite of, and replaces the `-nolib` option from prior releases. Previously the default behavior of `write_vhdl` was to output a separate VHDL file for each library used in the design, unless `-nolib` was specified. Now you must specify the `-lib` option to output separate files for each library.

`-port_diff_buffers` - (Optional) Add the differential pair buffers and internal wires associated with those buffers into the output ports list. This argument is only valid when `-mode pin_planning` or `-mode synth_stub` is specified.

`-write_all_overrides [ true | false ]` - (Optional) Write parameter overrides in the design to the VHDL output even if the value of the parameter is the same as the defined primitive default value. If the option is false then parameter values which are equivalent to the primitive defaults are not output to the VHDL file. Setting this option to true will not change the result but makes the output netlist more verbose.

`-keep_vcc_gnd` - (Optional) By default, when writing a nelist for simulation, or from an IP integrator block design, the Vivado Design Suite replaces VCC and GND primitives, and the nets they drive, with literal constants on each of the loads on the net. The `-keep_vcc_gnd` option disables this default behavior and preserves the VCC or GND primitives.

`-rename_top <arg>` - (Optional) Rename the top module in the output as specified. This option only works with `-mode funcsim` to allow the VHDL netlist to plug into top-level simulation test benches.

`-arch_only` - (Optional) Suppress the entity definition of the top module, and outputs the architecture only. This simplifies the use of the output VHDL netlist with a separate test bench.

Send Feedback

`-force` - (Optional) Overwrite the VHDL files if they already exists.

`-include_xilinx_libs` - (Optional) Write the simulation models directly in the output netlist file rather than pointing to the libraries by reference.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file>` - (Required) The filename of the VHDL file to write. If the file name does not have either a `.vhd` or `.vhdl` file extension then the name is assumed to be a directory, and the VHDL file is named after the top module, and is output to the specified directory.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

Examples

The following example writes a VHDL simulation netlist file for the whole design to the specified file and path:

```
write_vhdl C:/Data/bft_top.vhd
```

In the following example the entity definition of the top-level module is not output to the VHDL netlist:

```
write_vhdl C:/Data/vhdl_arch_only.vhd -arch_only
```

See Also

- write_verilog

write_waivers

Write out one or more DRC/METHODOLOGY/CDC message waivers in command form

Syntax

```
write_waivers [-type <arg>] [-objects <args>] [-return_string] [-force]
    [-quiet] [-verbose] [<file>]
```

Usage

Name	Description
[-type]	Type of waiver(s) - ALL, DRC, METHODOLOGY, CDC, LINT to write
[-objects]	List of DRC/METHODOLOGY/CDC waiver objects to be written
[-return_string]	Return report results as a string object
[-force]	Overwrite existing file
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<file>]	Name of file to write waivers

Categories

[Waiver](#), [Object](#)

Description

To save waivers from one design session to the next, you must use `write_waivers` to create an XDC file of the waiver commands, and `read_xdc` to read those waivers back into the design when it is reopened.

Arguments

`-type <arg>` - (Optional) Specifies the type of waivers to write to the file. Currently supports DRC, METHODOLOGY, and CDC.

`-objects <args>` - (Optional) Specifies a list of waiver objects to write. Waiver objects can be specified by the `get_waivers` command.

`-return_string` - (Optional) Directs the output to a Tcl string. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

`-force` - (Optional) This argument forces writing of waivers to the file. If you receive a message that the waivers will not be written because the counts will be invalid, you can use this option to write the file.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file>` - (Required) Specifies the file name to write.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

Examples

This example writes all waivers in the current design:

```
write_waivers C:/Data/design_waivers.xdc
```

The following example writes only DRC type waivers:

```
write_waivers -type DRC C:/Data/drc_waivers.xdc
```

See Also

- create_waiver
- delete_waivers
- get_cdc_violations
- get_drc_violations
- get_methodology_violations
- get_waivers
- report_cdc
- report_drc
- report_methodology
- report_waivers

Send Feedback

write_xdc

Writes constraints to a Xilinx Design Constraints (XDC) file. The default file extension for a XDC file is .xdc.

Syntax

```
write_xdc [-no_fixed_only] [-constraints <arg>] [-cell <arg>] [-sdc]
    [-no_tool_comments] [-force] [-exclude_timing] [-exclude_physical]
    [-add_netlist_placement] [-logic_function_stripped] [-type <args>]
    [-write_id] [-exclude_cells <args>] [-quiet] [-verbose] [<file>]
```

Usage

Name	Description
[-no_fixed_only]	Export fixed and non-fixed placement (by default only fixed placement is exported)
[-constraints]	Include constraints that are flagged invalid Values: valid, invalid, all Default: valid
[-cell]	Hierarchical cell for which constraints are exported.
[-sdc]	Export all timing constriants in SDC compatible format.
[-no_tool_comments]	Don't write verbose tool generated comments to the xdc when translating from ucf.
[-force]	Overwrite existing file.
[-exclude_timing]	Don't export timing constraints.
[-exclude_physical]	Don't export physical constraints.
[-add_netlist_placement]	Export netlist placement constraints.
[-logic_function_stripped]	Write disable_timing constraints which are associated with having previously run write_edif with its -logic_function_stripped option.
[-type]	Types of constraint to export. Values: timing, io, misc, waiver and physical. If not specified, all constraints will be exported.
[-write_id]	Write position number for timing constraints
[-exclude_cells]	exclude constraints scoped to <cell(s)> or to any hierarchical module below <cell(s)>
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[<file>]	Output constraints to the specified XDC file.

Categories

[Timing](), [FileIO]()

Description

Writes constraints to a Xilinx® Design Constraints file (XDC). The XDC can be exported from the top-level, which is the default, or from a specific hierarchical cell.

Send Feedback

> ⭐ **IMPORTANT!** *The `write_xdc` command writes the constraints to the specified file in the same order they are added to or executed in the design.*

The `write_xdc` command lets you write invalid XDC constraints so that you can quickly report constraints that have been ignored by the Vivado Design Suite due to a problem with the way the constraint is written or applied. This is useful for debugging constraint files applied in specific designs.

This command can be used to create an XDC file from a design with UCF files. All constraints from the active constraint fileset will be exported to the XDC, even if they come from multiple files.

> 💡 **TIP:** *The `write_xdc` command will not convert all UCF constraints into XDC format, and is not intended to automatically convert UCF based designs to XDC. Refer to the Vivado Design Suite Migration Methodology Guide (UG911) for more information on migrating UCF constraints to XDC.*

Arguments

`-no_fixed_only` - (Optional) Export both fixed and unfixed placement LOCs to the constraint file being written. By default only the fixed LOCs will be written to the XDC file. Fixed LOCs are associated with user-assigned placements, while unfixed LOCs are associated with tool assigned placements.

`-constraints <arg>` - (Optional) Export constraints that are flagged valid, invalid, or all constraints (both valid and invalid). The default behavior is to export only valid constraints to the XDC file. Valid values are VALID, INVALID, or ALL.

`-cell <arg>` - (Optional) The name of a hierarchical cell in the current design to export the constraints from. The constraints will be written to the specified XDC file relative to the specified cell.

Note: A design must be open when using this option.

`-exclude_cells <arg>` - (Optional) The name of a hierarchical cell in the current design to exclude the constraints from. Only constraints scoped to the specified cell or to any hierarchical module below the cell are excluded from the written XDC. The top-level constraints are not affected, even when pointing to objects below the specified cell.

`-sdc` - (Optional) Export only the timing constraints in a file format that is 100% SDC compatible from the current design. Does not export any other defined constraints.

`-no_tool_comments` - (Optional) Do not add tool generated comments into the XDC file.

`-force` - (Optional) Overwrite a file of the same name if one already exists.

`-exclude_timing` - (Optional) Do not export timing constraints. This results in an XDC file that contains only physical constraints.

`-exclude_physical` - (Optional) Do not export physical constraints. This results in an XDC file that contains only timing constraints.

`-add_netlist_placement` - (Optional) Include placement constraints that are defined in the netlist file as part of the written XDC file.

`-type <arg>` - (Optional) Specifies the types of constraint to export. Valid values are: timing, io, misc, waiver and physical. Multiple types can be specified at one time. If the type is not specified, all constraints will be exported.

`-logic_function_stripped` - (Optional) Includes the `set_disable_timing` constraints which are needed to preserve the constants propagation and timing signoff after stripping the logic function of the LUTs. It should be used in association with `write_edif -logic_function_stripped`.

`-write_id` - (Optional) Includes the constraint position number before each timing constraint. The position number if also reported inside the Timing Constraints Editor (TCE) and some of the Methodology violations.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<file>` - (Required) The file name of the XDC file to write.

Note: If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

Examples

Write the valid and invalid constraints, including both fixed and unfixed cells, to the specified file.

```
write_xdc -no_fixed_only -constraints all C:/Data/design.xdc
```

Write only the invalid constraints, including both fixed and unfixed cells, to the specified file.

```
write_xdc -constraints invalid C:/Data/bad_constraints.xdc
```

Write the physical constraints only, including placement constraints defined in any netlist source files.

```
write_xdc -exclude_timing -add_netlist_placement C:/Data/physical.xdc
```

See Also

- read_xdc

Send Feedback

write_xsim_coverage

Dumps the current coverage database into the disk. A subsequent export_xsim_coverage should generate the corresponding coverage report.

Syntax

```
write_xsim_coverage [-cov_db_name <arg>] [-cov_db_dir <arg>] [-quiet]
     [-verbose]
```

Usage

Name	Description
`[-cov_db_name]`	Specify the coverage database that is to be saved. Default: <snapshot_name>, default snapshot_name is work.<simulation_top>
`[-cov_db_dir]`	The output coverage database directory where the database will be saved. Default : .
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution

Categories

Simulation

Send Feedback

xsim

Load a simulation snapshot for simulation and return a simulation object

Syntax

```
xsim [-view <args>] [-autoloadwcfg] [-runall] [-R] [-maxdeltaid <arg>]
    [-nolog] [-maxlogsize <arg>] [-onfinish <arg>] [-onerror <arg>]
    [-tclbatch <args>] [-t <args>] [-testplusarg <args>] [-vcdfile <arg>]
    [-vcdunit <arg>] [-wdb <arg>] [-tp] [-tl] [-nosignalhandlers]
    [-ieeewarnings] [-stats] [-scNoLogFile] [-sv_seed <arg>]
    [-protoinst <args>] [-cov_db_dir <arg>] [-cov_db_name <arg>]
    [-ignore_assertions] [-ignore_coverage] [-downgrade_error2info]
    [-downgrade_error2warning] [-downgrade_fatal2info]
    [-downgrade_fatal2warning] [-ignore_feature <args>]
    [-downgrade_severity <args>] [-cosim_hostname <arg>]
    [-cosim_portno <arg>] [-quiet] [-verbose] <snapshot>
```

Returns

Current simulation object.

Usage

Name	Description
[-view]	Open a wave configuration file. This switch may be repeated to open multiple files.
[-autoloadwcfg]	For a WDB file named <name>.wdb, automatically open all WCFG files named <name>#.wcfg. Ignored if -view is present.
[-runall]	Run simulation until completion, then quit (does 'run -all; exit')
[-R]	Run simulation until completion, then quit (does 'run -all; exit')
[-maxdeltaid]	Specify the maximum delta number. Will report error if it exceeds maximum simulation loops at the same time Default: 10000
[-nolog]	Ignored (for compatibility with xsim command-line tool)
[-maxlogsize]	Set the maximum size a log file can reach in MB. The default setting is unlimited Default: -1
[-onfinish]	Specify behavior at end of simulation: quit|stop Default: stop
[-onerror]	Specify behavior upon simulation run-time error: quit|stop Default: stop
[-tclbatch]	Specify the TCL file for batch mode execution
[-t]	Specify the TCL file for batch mode execution
[-testplusarg]	Specify plusargs to be used by $test$plusargs and $value $plusargs system functions
[-vcdfile]	Specify the vcd output file

Send Feedback

Name	Description
`[-vcdunit]`	Specify the vcd output unit. Default is the same as the engine precision unit
`[-wdb]`	Specify the waveform database output file
`[-tp]`	Enable printing of hierarchical names of process being executed
`[-tl]`	Enable printing of file name and line number of statements being executed.
`[-nosignalhandlers]`	Run with no signal handlers to avoid conflict with security software
`[-ieeewarnings]`	Enable warnings from VHDL IEEE functions
`[-stats]`	Display memory and cpu stats upon exiting
`[-scNoLogFile]`	Keep the SystemC output separate from XSim output
`[-sv_seed]`	Seed for constraint random stimulus Default: 1
`[-protoinst]`	Specify a .protoinst file for protocol analysis
`[-cov_db_dir]`	System Verilog Coverage Run Directory. The coverage data will be present under <cov_db_dir>/xsim.covdb/ <cov_db_name> directory.Default is ./ or inherits value set in similar xelab option.
`[-cov_db_name]`	System Verilog Coverage Run Name. The coverage data will be present under <cov_db_dir>/xsim.covdb/<cov_db_name> directory.Default is snapshot name or inherits value set in similar xelab option.
`[-ignore_assertions]`	Ignore System Verilog concurrent assertion constructs at runtime.
`[-ignore_coverage]`	Ignore System Verilog Functional Coverage at runtime.
`[-downgrade_error2info]`	Downgrade System verilog message severity from error to info level.
`[-downgrade_error2warning]`	Downgrade System verilog message severity from error to warning level.
`[-downgrade_fatal2info]`	Downgrade System verilog message severity from fatal to info level.
`[-downgrade_fatal2warning]`	Downgrade System verilog message severity from fatal to warning level.
`[-ignore_feature]`	Specify System Verilog feature to be ignored at runtime. Choices are: assertion : Ignore concurrent assertions.
`[-downgrade_severity]`	Downgrade severity level of System Verilog HDL messages.Choices are: error2warning|error2info| fatal2warning|fatal2info.
`[-cosim_hostname]`	Hw Cosim Hw Server Name
`[-cosim_portno]`	Hw Cosim Hw Port Number
`[-quiet]`	Ignore command errors
`[-verbose]`	Suspend message limits during command execution
`<snapshot>`	The name of design snapshot or WDB file

Categories

[Simulation](#)

Description

The `xsim` command loads a simulation snapshot to run a batch mode simulation, or to provide a GUI and/or Tcl-based interactive simulation environment. The snapshot must be generated using the `xelab` command.

Arguments

`-view <arg>` - (Optional) Open a wave configuration file to store the waveform activity for the simulation. The Wave Config file contains just the list of wave objects (signals, dividers, groups, virtual buses) to display, and their display properties, plus markers. A wave configuration file is written in the current simulation with the `save_wave_config` command.

Note: This option may be repeated to open multiple Wave Config files.

`-autoloadwcfg` - (Optional) When loading a wave database (WDB) file named `<name>.wdb`, automatically open all associated wave config files WCFG named `<name>#.wcfg`. This option is ignored if `-view` is also specified.

`-runall | -R` - (Optional) Run the simulation until no event is left in the event queue, a breakpoint or valid condition is encountered, or a run time exception occurs. Then quit the simulator. This is the equivalent of 'run -all; exit'.

`-maxdeltaid <arg>` - (Optional) Specify the maximum number of delta cycles as an integer greater than 0. The default value is 10000.The simulator will report an error if it exceeds the specified maximum number of simulation loops, or delta cycles, at simulation run time. Refer to the *Vivado Design Suite User Guide: Logic Simulation (UG900)* for more information on delta cycles.

`-nolog` - (Optional) This option is provided for compatibility with the command line XSIM utility, and is ignored when running in Tcl in the Vivado Design Suite.

`-maxlogsize <arg>` - (Optional) The maximum size for a simulation log file, specified as a value in MBytes. The default value of -1, means the log file has no size limit.

`-onfinish [ stop | quit ]` - (Optional) Specify the actions of the simulator upon finishing the simulation run. Valid values are stop the simulation, or quit the simulator. The default is stop.

`-onerror [ stop | quit ]` - (Optional} Specify the actions of the simulator upon encountering an error. Valid values are stop the simulation, or quit the simulator. The default is stop.

`-tclbatch | -t` - (Optional) Specify a Tcl script file to run the simulator in batch mode.

`-testplusarg <args>` - (Optional) Specify plusargs to be used by `$test$plusargs` and `$value$plusargs` system functions. These arguments are visually distinguished from other simulator arguments by starting with the plus ('+') character.

`-vcdfile <arg>` - (Optional) Specify a Value Change Dump (VCD) file to capture simulation output.

`-vcdunit <arg>` - (Optional) Specify the time unit for the VCD output. The default unit is the same as the simulation engine precision.

`-wdb <arg>` - (Optional) Specify the simulation waveform database (WDB) file. When the simulation completes, the simulation is written to a static simulator database file. The file can be opened for later review by the `open_wave_database` command.

`-tp` - (Optional) Print the hierarchical names of process being executed to the standard output.

`-tl` - (Optional) Print the file name and line number of statements being executed to the standard output.

`-nosignalhandlers` - (Optional) Disables the installation of OS-level signal handlers in the simulation. With the signal handlers disabled, OS-level fatal errors could crash the simulation abruptly with little indication of the nature of the failure.

> ⭐ **IMPORTANT!** *Use this option only if your security software prevents the simulator from running successfully.*

`-ieeewarnings` - (Optional) Enable warnings generated by the use of VHDL IEEE functions. Use this option to enable warnings that are disabled by default since these warnings can generally be ignored.

`-stats` - (Optional) Display statistics related to memory and CPU usage upon exiting the simulator.

`-quiet` - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note: Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

`-verbose` - (Optional) Temporarily override any message limits and return all messages from this command.

Note: Message limits can be defined with the `set_msg_config` command.

`<snapshot>` - (Required) The name of the simulation snapshot to be executed, or WDB file to be opened for viewing. The snapshot must have been previously compiled by `xelab`. The WDB file must have been previously saved using the `-wdb` option of the `xsim` command.

Examples

The following example launches xsim on the specified simulation snapshot:

```
xsim C:/Data/project_xsim/project_xsim.sim/sim_1/behav/testbench_behav
```

See Also

- launch_simulation

Send Feedback

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Xilinx Support.

Documentation Navigator and Design Hubs

Xilinx® Documentation Navigator (DocNav) provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open DocNav:

- From the Vivado® IDE, select **Help→Documentation and Tutorials**.
- On Windows, select **Start→All Programs→Xilinx Design Tools→DocNav**.
- At the Linux command prompt, enter `docnav`.

Xilinx Design Hubs provide links to documentation organized by design tasks and other topics, which you can use to learn key concepts and address frequently asked questions. To access the Design Hubs:

- In DocNav, click the **Design Hubs View** tab.
- On the Xilinx website, see the Design Hubs page.

Note: For more information on DocNav, see the Documentation Navigator page on the Xilinx website.

Send Feedback

Tcl Resources

Training Resources

Xilinx provides a variety of training courses and QuickTake videos to help you learn more about the concepts presented in this document. Use these links to explore related training resources:

- UltraFast Design Methodology Training Course
- Designing with UltraScale and UltraScale+ Architectures Training Course
- Designing FPGAs Using the Vivado Design Suite Training Course
- Vivado Design Suite QuickTake Video: Using the Non-Project Batch Flow
- Vivado Design Suite QuickTake Video: Using Tcl Scripts as Constraint Files in Vivado

References

- *Vivado Design Suite User Guide: Design Flows Overview* (UG892)
- *Vivado Design Suite User Guide: Using the Vivado IDE* (UG893)
- *Vivado Design Suite User Guide: Using Tcl Scripting* (UG894)
- *Vivado Design Suite User Guide: Using Constraints* (UG903)
- *Vivado Design Suite Properties Reference Guide* (UG912)

Tcl Developer Xchange

Tcl reference material is available on the Internet. Xilinx recommends the Tcl Developer Xchange, which maintains the open source code base for Tcl, and is located at:

http://www.tcl.tk

An introductory tutorial is available at:

http://www.tcl.tk/man/tcl8.5/tutorial/tcltutorial.html

About SDC

Synopsys Design Constraints (SDC) is an accepted industry standard for communicating design intent to tools, particularly for timing analysis. A reference copy of the SDC specification is available from Synopsys by registering for the TAP-in program at:

http://www.synopsys.com/Community/Interoperability/Pages/TapinSDC.aspx

Please Read: Important Legal Notices

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at https://www.xilinx.com/legal.htm#tos; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at https://www.xilinx.com/legal.htm#tos.

AUTOMOTIVE APPLICATIONS DISCLAIMER

AUTOMOTIVE PRODUCTS (IDENTIFIED AS "XA" IN THE PART NUMBER) ARE NOT WARRANTED FOR USE IN THE DEPLOYMENT OF AIRBAGS OR FOR USE IN APPLICATIONS THAT AFFECT CONTROL OF A VEHICLE ("SAFETY APPLICATION") UNLESS THERE IS A SAFETY CONCEPT OR REDUNDANCY FEATURE CONSISTENT WITH THE ISO 26262 AUTOMOTIVE SAFETY STANDARD ("SAFETY DESIGN"). CUSTOMER SHALL, PRIOR TO USING OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE PRODUCTS, THOROUGHLY TEST SUCH SYSTEMS FOR SAFETY PURPOSES. USE OF PRODUCTS IN A SAFETY APPLICATION WITHOUT A SAFETY DESIGN IS FULLY AT THE RISK OF CUSTOMER, SUBJECT ONLY TO APPLICABLE LAWS AND REGULATIONS GOVERNING LIMITATIONS ON PRODUCT LIABILITY.

Copyright