See all versions of this document

Vivado Design Suite User Guide

I/O and Clock Planning

UG899 (v2021.2) November 10, 2021

Xilinx is creating an environment where employees, customers, and partners feel welcome and included. To that end, we're removing non-inclusive language from our products and related collateral. We've launched an internal initiative to remove language that could exclude people or reinforce historical biases, including terms embedded in our software and IPs. You may still find examples of non-inclusive language in our older products as we work to make these changes and align with evolving industry standards. Follow this link for more information.

XILINX

Revision History

The following table shows the revision history for this document.

Section	Revision Summary
11/10/2021 Version 2021.2	
Chapter 6: Hard Block Planner	New Chapter

Table of Contents

Revision History 2

Chapter 1: Introduction 5
- Navigating Content by Design Process 5
- I/O and Clock Planning Stages 7
- I/O and Clock Planning Using the Platform Board Flow 15
- I/O and Clock Planning for SSI Technology Devices 16
- I/O and Clock Planning for IP with I/O Ports 16
- I/O Planning for Zynq UltraScale+ MPSoCs 17
- I/O Planning for UltraScale and UltraScale+ 17
- I/O Planning for Versal ACAP 17

Chapter 2: Configuring the Device 19
- Defining Alternate Compatible Parts 19
- Setting Device Configuration Modes 20
- Setting Device Constraints 23
- Setting the Configuration Bank Voltage Select Pin 27

Chapter 3: I/O Logic and Low-Speed I/O Planning 29
- Using the I/O Planning View Layout 30
- Viewing Device Resources 31
- Defining and Configuring I/O Ports 35
- Placing I/O Ports 47
- Migrating an I/O Planning Project to an RTL Project 53

Chapter 4: I/O Planning for UltraScale Architecture Memory IP 54
- UltraScale Architecture Memory IP I/O Planning Design Flow Changes 55
- Configuring Memory IP 56
- UltraScale Architecture Memory IP I/O Planning in the Vivado IDE 58
- Using the Memory Bank/Byte Planner 59
- Modifying Memory I/O Ports 68
- Running Memory DRCs 70
- Implementing the PHY 70

Copying I/O Port Assignments Between Designs....71

Chapter 5: High-Speed I/O Planning for Versal ACAP 73
Using the Advanced IO Planner.... 73

Chapter 6: Hard Block Planner....78
Hard Block Planning for Versal ACAP.... 78
Using the Hard Block Planner....78

Chapter 7: Clock Planning....86
Locating Logic Cells....87
Placing Clock Logic in the Device Window.... 87

Chapter 8: Validating I/O and Clock Planning.... 88
Running DRCs.... 88
Working with SSN Analysis....94

Chapter 9: Interfacing with the System Designer.... 101
Exporting I/O Pin and Package Data.... 101
Generating IBIS Models.... 102
Interfacing with the PCB Design.... 104

Appendix A: Using I/O Port Lists in CSV File Format.... 107
CSV File.... 107
Differential Pairs in the CSV File.... 110

Appendix B: Additional Resources and Legal Notices.... 111
Xilinx Resources....111
Solution Centers.... 111
Documentation Navigator and Design Hubs.... 111
References....112
Training Resources....113
Please Read: Important Legal Notices.... 114

Chapter 1

Introduction

I/O and clock planning is the process of defining and analyzing the connectivity between the FPGA/ACAP and the printed circuit board (PCB) and assigning the various interconnect signals to physical pins of the device. This process includes PCB designers, FPGA designers, and system designers with the following concerns and requirements:

- Streamlining the critical signal connectivity to shorten signal lengths and avoid signal crossings.
- Maintaining the integrity of high speed signals on and off of the device.
- Selecting an I/O configuration that might work with alternate devices.
- Determining power and ground signal availability on the PCB.
- Establishing PCB requirements for proper decoupling.
- Identifying device programming and debugging considerations.

Often, designers are hindered by a non-optimal pinout that causes further delays when trying to meet timing and signal integrity requirements. By considering the data flow from the PCB to the FPGA/ACAP die, you can achieve optimal pinout configurations quickly, thus reducing internal and external trace lengths as well as routing congestion. This chapter provides an overview of the I/O and clock planning process using the graphical user interface (GUI) known as the Vivado® Integrated Design Environment (IDE).

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 and the Design Flow Assistant materials can be found on the Xilinx.com website. This document covers the following design processes:

- **System and Solution Planning:** Identifying the components, performance, I/O, and data transfer requirements at a system level. Includes application mapping for the solution to PS, PL, and AI Engine.

The following table summarizes various interface speed options available in each architecture for easy document navigation. Note that the bit-rates mentioned in the table are based on hardware characterization using LVDS in speed grade -3 devices.

Table 1: **Navigating IP and Clock Planning Content**

Device Architecture	Additional Considerations	Relevant links
7 series	**HRIO (Low Speed I/O)** • Bit-rate range 0 to 1250 Mb/s **HPIO (High Speed I/O)** • Bit-rate range 0 to 1600 Mb/s	• Chapter 3: I/O Logic and Low-Speed I/O Planning • Chapter 7: Clock Planning • Chapter 8: Validating I/O and Clock Planning • Chapter 9: Interfacing with the System Designer
UltraScale/UltraScale+	**Component Mode (Low Speed I/O)** • Bit-rate range for UltraScale HP/HR bank or UltraScale+ HP bank 0-1250 Mb/s • Bit-rate range for UltraScale+ HD bank 0-250 Mb/s	• Chapter 3: I/O Logic and Low-Speed I/O Planning • Chapter 7: Clock Planning • Chapter 8: Validating I/O and Clock Planning • Chapter 9: Interfacing with the System Designer
	Native Mode (High Speed I/O) • Bit-rate range 300 to 1600 Mb/s (only HP banks)	• Chapter 4: I/O Planning for UltraScale Architecture Memory IP • Chapter 7: Clock Planning • Chapter 8: Validating I/O and Clock Planning • Chapter 9: Interfacing with the System Designer

Table 1: **Navigating IP and Clock Planning Content** *(cont'd)*

Device Architecture	Additional Considerations	Relevant links
Versal ACAP	**I/O Logic (Low Speed I/O)** • Bit-rate range 0-300 Mb/s (Both HD and XP banks)	• *Versal ACAP SelectIO Resources Architecture Manual* (AM010) • Chapter 7: Clock Planning • Chapter 8: Validating I/O and Clock Planning • Chapter 9: Interfacing with the System Designer
	XPIO (High Speed I/O) • Bit-rate range 200-1800 Mb/s (Only XP banks)	• Chapter 5: High-Speed I/O Planning for Versal ACAP • Chapter 7: Clock Planning • Chapter 8: Validating I/O and Clock Planning • Chapter 9: Interfacing with the System Designer

I/O and Clock Planning Stages

The Vivado Design Suite facilitates I/O and clock planning at different stages of the design process from initial collaboration between the PCB designer and the FPGA designer to validation of a fully implemented design. As the design progresses through the design flow, more information becomes available, which enables more complex analysis and rule checking. For example, analysis early in the design flow (synthesis/opt) uses actual cell delays but zero interconnect delay. Post place the cell delays are actual and interconnect delays are estimated, while the implemented design uses actual cell and interconnect delays for routed nets.

Proper I/O assignment depends on the structure of the FPGA, the requirements of the PCB design, and the interaction between the two. Visualizing how the FPGA interacts logically and physically with the PCB enables streamlining of the data flow through the device. I/O port assignment, which defines how signals from the PCB come into the FPGA design or go out to the board, and clock resource assignment, which defines the structure of the clock tree in the design, are often completed together.

For example, certain pins on the device are optimal for clock pins while others are optimal for digitally controlled impedance (DCI) cascade and internal voltage reference (VREF).

Failing to properly plan the I/O port and clock assignments can lead to decreased system performance, multiple design iterations, and longer design closure times. Versal clock and I/O design is easy with the Advanced I/O wizard setup I/O and clocking through a simple GUI. For information on board and device planning using the UltraFast™ design methodology, see this link in the *UltraFast Design Methodology Guide for Xilinx FPGAs and SoCs* (UG949).

You can perform I/O planning at any stage in the design flow. For example, you can begin the I/O assignment process from a top-level ports list, a register transfer level (RTL) design, or a synthesized netlist. Various types of projects facilitate flexible entry points for I/O planning. Whenever possible, it is best to perform I/O assignment with a synthesized design. For example, you can only perform more complex I/O placement design rule checks (DRCs) on a synthesized design.

I/O Planning can be done in a number of different ways. After building a design in the Advanced I/O wizard, there is a new tool for Versal called "Advanced I/O Planner" that allows pin planning for any SelectIO or soft/hard memory interface on a bank (54 pins) and/or nibble level (6 pins) granularity. In the Advanced I/O Planner, you can autoplace your pin assignments, then adjust individual pin assignments through the classic pin planning tool. This tool can autoplace all the I/O interfaces to maximize the clocking and I/O architecture. If you need to place an individual I/O, the classic pin planning tools that write out pin constraints to an XDC file are still supported. Finally, you can also design your pin plan with a user-defined XDC file.

Certain types of IP, such as Memory IP, gigabit transceivers (GT), Xilinx® High Speed IO IP, PCI Express® (PCIe), and Ethernet interfaces have I/O ports associated with them. You must properly configure this IP using the IP capabilities in the Vivado Design Suite prior to beginning the I/O planning process. Because these interfaces are usually the most timing critical, use this IP as the starting point when considering the device pin assignments. In addition, use an RTL or synthesized design when using this IP. For details, see I/O and Clock Planning for IP with I/O Ports.

Related Information

I/O and Clock Planning for IP with I/O Ports

I/O and Clock Planning Design Flow

In the Vivado Design Suite, you can work on I/O and clock planning at any stage in the design flow using any type of project. Following are the most commonly used methods.

TIP: *You can also run the Vivado Design Suite and perform I/O and clock planning for Non-Project Mode. For information on both Project Mode and Non-Project Mode, see this link in the Vivado Design Suite User Guide: Design Flows Overview (UG892).*

VIDEO: *For more information on performing I/O planning at various stages of the design process see the Vivado Design Suite QuickTake Video: I/O Planning Overview.*

Pre-RTL I/O Planning

You can create an empty I/O planning project to enable early device exploration and initial I/O port assignment before the design source files are available. With this method, you do not have RTL source files or a netlist, and you are working on initial I/O planning and board-level integration. This enables PCB and FPGA designers to agree on an early pinout definition, which can eliminate iterations related to device pinout changes later in the design cycle. Using the I/O planning project, you can:

- Import device and I/O port assignments from the PCB designer or create I/O ports manually.
- Export device and I/O port assignments to hand off to the PCB designer or for use later in the design process.
- Migrate an I/O planning project to an RTL project when the port definitions and pin assignments are resolved.
- Create a Verilog or VHDL module definition for the top-level of the design based on your port definitions.

  When you complete the port assignments in the I/O planning project, you can migrate the project to an RTL project and create the Verilog or VHDL module definition for the top-level of the design. This allows you to use the agreed upon I/O plan as the start of your RTL design. For more information, see Migrating an I/O Planning Project to an RTL Project.

  ***Note*:** For information on creating an I/O planning project, see this link in the *Vivado Design Suite User Guide: System-Level Design Entry* (UG895). For information on importing pin assignments defined by the PCB designer or from another Vivado Design Suite project, see Defining and Configuring I/O Ports.

Related Information

Migrating an I/O Planning Project to an RTL Project
Defining and Configuring I/O Ports

RTL I/O Planning

You can perform I/O planning in an elaborated RTL project. With this method, you use an RTL design, which optionally includes IP cores from the Vivado IP catalog or block designs from the Vivado IP integrator. Using the IP catalog, you can customize IP, customize clocking components using the Clocking wizard, and configure SelectIO™ interface resources using the SelectIO Interface wizard. In an elaborated design, the Vivado tools provide basic DRCs to check port assignments, I/O standards, clock resources, and other design details. You can do initial I/O and clock planning with the elaborated design and export device and I/O port assignments for use in PCB schematic symbol generation or save the constraints in an XDC file for use during synthesis or implementation.

***Note*:** For information on creating an RTL project and opening the elaborated design, see this link in the *Vivado Design Suite User Guide: System-Level Design Entry* (UG895).

Netlist I/O Planning

You can also perform I/O planning with a synthesized netlist. With this method, you use either a synthesized RTL project or a netlist project created with a post-synthesis netlist. Whenever possible, use a synthesized design to perform I/O and clock planning. The Vivado tools have more information about the design after synthesis, and you can use automatic I/O placement and interactive placement modes to control I/O port assignment. You can also use the I/O Planning view layout to see the relationship between the physical pins of the device package and the die pads of the I/O banks on the device.

Using a synthesized design also enables you to make more informed decisions when optimizing the connectivity between the PCB and the Xilinx device. This allows you to better interface with the PCB or system-level designer, making it easier to incorporate IO placement from IP cores that assign I/O placement, like MIPI or the Memory IP. In addition, because all clocks, including generated clocks, are defined after synthesis, the Vivado Design Suite has greater visibility into the clocking requirements and resource utilization and can perform a more thorough validation of the design.

Note: You can perform netlist-based I/O planning on a synthesized RTL design or on a post-synthesis netlist project. For information on creating a post-synthesis project, see this link in the *Vivado Design Suite User Guide: System-Level Design Entry* (UG895).

RECOMMENDED: *To check clock logic, Xilinx recommends validation with a synthesized design. To check clock timing, Xilinx recommends validation with an implemented design.*

Final I/O Validation with an Implemented Design

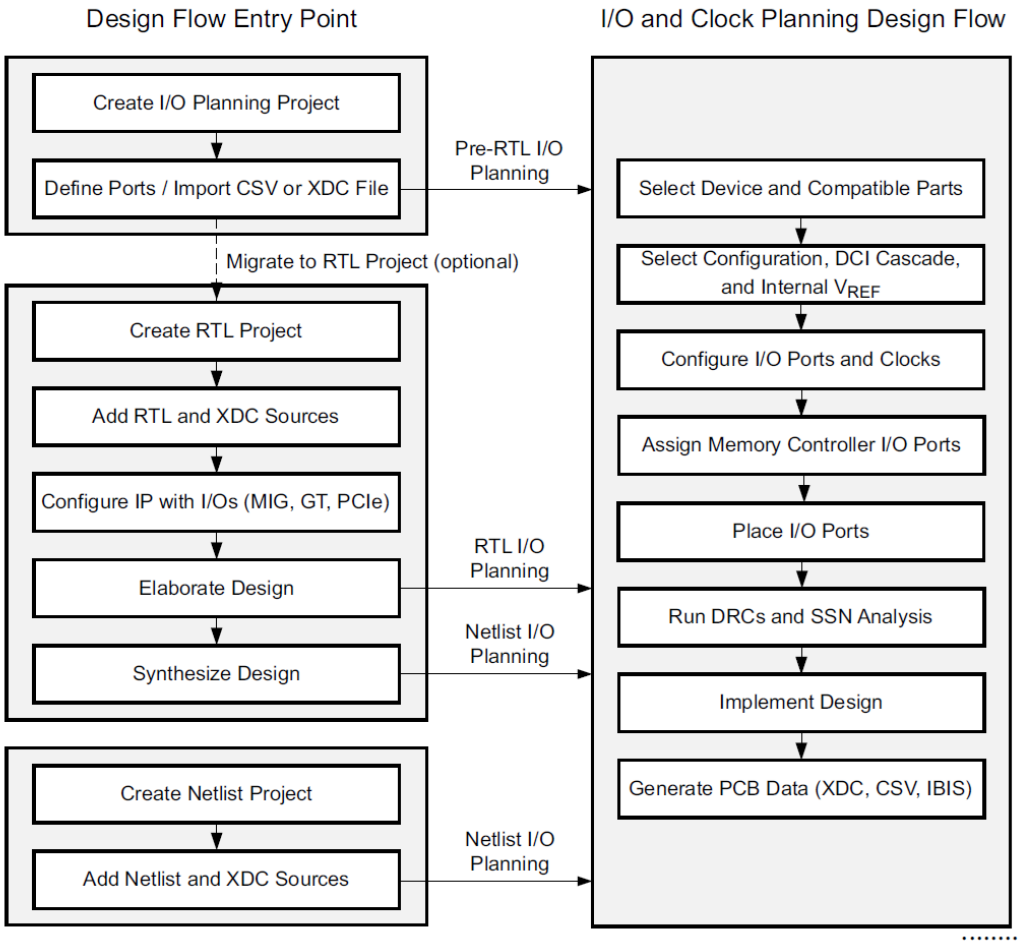
You must use a fully implemented design to validate the final valid I/O pinout and clock configuration. Proper clock resource validation requires fully routed implementation of all clocks. You can examine the implementation reports for I/O and clock-related messages. Finally, double-check the I/O port assignments with the PCB designer to ensure that the FPGA is correctly defined for the system-level design.

Note that I/O placement and location is highly dependent on the clock placement and what nibble the clock comes into. See *Versal ACAP SelectIO Resources Architecture Manual* (AM010) and *Versal ACAP Clocking Resources Architecture Manual* (AM003) for more information about clocking in I/O.

I/O and Clock Planning Design Flow Steps

The following figure shows the project design flow steps on the left. Horizontal arrows indicate the points in the project design flow when you can perform I/O and clock planning. The steps in the I/O and clock planning design flow are shown on the right.

Figure 1: **I/O and Clock Planning Design Flow**



The project design flow starts with an empty I/O planning project, an RTL design project, or a post-synthesis netlist project. Using any of these project types, you can perform the following steps in the I/O and clock planning design flow:

1. Select Device and Compatible Parts

   When selecting the part, determine the device size based on resource estimates for the final design. Select the package based on PCB requirements, such as critical routes to memories. Versal also has dedicated hard memory controller pins that cannot be used for other I/O operations. For designs using stacked silicon interconnect (SSI) technology, see this link in the *UltraFast Design Methodology Guide for Xilinx FPGAs and SoCs* (UG949). In addition to the selected part, you can also identify alternate compatible parts, as described in Defining Alternate Compatible Parts.

Alternatively, you can select a supported targeted design platform board, which includes a Xilinx device and other components to provide a robust evaluation platform or rapid product development platform. For more information, see I/O and Clock Planning Using the Platform Board Flow.

2. Select Configuration, DCI Cascade, and Internal VREF

   A Xilinx device must be configured each time it is powered up. The bitstream is loaded into the device through special configuration pins that enable different configuration modes. For Versal, a device image is loaded. The configuration modes used in an application can affect the I/O planning of the design. It is important to determine and plan for the configuration modes before beginning I/O assignment. The configuration mode not only determines the connectivity of certain pins, it also determines the VCCO voltage required for the I/O bank that includes multi-function pins. For information, see Setting Device Configuration Modes.

   Depending on the I/O standard, digitally controlled impedance (DCI) can control the output impedance of a driver or add a parallel termination to the driver, receiver, or both to match the characteristic impedance of a transmission line and improve signal integrity. DCI uses two multi-purpose reference pins in each I/O bank to control the impedance of the driver or the parallel-termination value for all of the I/Os in the bank.

   Single-ended I/O standards with a differential input buffer require reference voltage (VREF). You can generate an internal VREF using the INTERNAL_VREF constraint, which eliminates the need to provide a particular reference voltage supply rail on the PCB. In 7 series and UltraScale™ architecture, this can free multi-purpose VREF pins in a given I/O bank for other I/O port assignments. For more information, see Setting Device Constraints.

3. Configure I/O Ports and Clocks

   The I/O ports on the device support multiple I/O-related constraints, such as IOSTANDARD, SLEW, and DRIVE. Configure these ports to support the standards required for the system-level design. The I/O standard definition might impact pin placement. For example, you can combine some I/O standards within a single I/O bank but not others. For more information, see Configuring I/O Ports.

   Xilinx devices are subdivided into columns and rows of clock regions. A clock region contains configurable logic blocks (CLBs), I/O banks, digital signal processing (DSP) slices, block random access memory (RAM), interconnect, and associated clocking resources. Each I/O bank contains clock-capable input pins to bring system or board clocks onto the device and into clock routing resources. You must plan the use of these clock resources to distribute the clocks in your design across your device. Versal has specific clocking for high speed I/O that does not go out on global clocking. Be careful to ensure that you correctly clock these I/O. For more information, see Clock Planning.

   Note: You cannot perform clock planning in an I/O planning project, because clock objects are not defined in this type of project.

RECOMMENDED: *Xilinx recommends that you use the Clocking wizard in the Vivado IP catalog to generate mixed-mode clock manager (MMCM) or phase-locked loop (PLL) modules to define clock connections. For information, see the Clocking Wizard LogiCORE IP Product Guide (PG065). You can also generate clocking with an I/O interface design through the Advanced I/O Wizard.*

4. Assign Memory Controller I/O Ports

   The Memory IP defines a memory controller using a pre-engineered controller and physical layer (PHY) for interfacing FPGA designs to supported external memory devices. High-speed memory controllers as well as Ethernet IP and PCI Express® (PCIe) technology IP have specific pinout requirements driven by clocking and skew needs.

   You must define the I/O physical pin assignments for the gigabit transceiver (GT), PCIe technology, and Memory IP as part of IP customization when the core is added to the design. Depending on the IP, to change the I/O assignments, you must re-customize the IP in the design. For information on working with and customizing IP, see the *Vivado Design Suite User Guide: Designing with IP* (UG896). For UltraScale architecture Memory IP, the I/O assignment is integrated into the standard I/O planning flow and does not require Memory IP customization. For more information, see I/O Planning for UltraScale Architecture Memory IP.

   Note: I/O planning projects do not read the physical pin assignments from the IP files for complex IP like the Memory Controllers, PCIe, or gigabit transceivers. See I/O and Clock Planning for IP with I/O Ports for more information.

5. Place I/O Ports

   You can interactively assign I/O ports in the design to package pins on the device using different methods. You can select individual I/O ports or groups of I/O ports called interfaces in the I/O Ports window and assign them to package pins in the Package window or to I/O pads in the Device window. The Advanced I/O Planner is also a tool for nibble and bank level pin placement. This tool allows for automatic placement that can understand all the I/O interfaces in an XPIO bank and place them effectively all at once. For information, see Placing I/O Ports.

   You can also let the Vivado Design Suite automatically place I/O ports using information derived from the synthesized design. For information, see Automatically Placing I/O Ports.

6. Run DRCs and SSN Analysis

   After making I/O and clock assignments, it is critical that you analyze your design by running design rule checks (DRCs) and simultaneous switching noise (SSN) analysis. DRCs validate the current design against a specified set of design rules and report any violations. For information, see Running DRCs.

   SSN analysis estimates the disruption that simultaneously switching outputs can cause on other output ports in an I/O bank. The calculation and estimates incorporate I/O bank-specific electrical characteristics into the prediction to identify potential noise-related issues in your design. For information, see Working with SSN Analysis.

Note: SSN analysis estimates are intended to identify potential noise-related issues in your design and are not intended as final design sign-off criteria.

RECOMMENDED: *Xilinx recommends that you run DRCs and SSN analysis after synthesis, prior to implementation, as well as after implementation. This enables you to catch issues early in the design cycle.*

7. Implement Design

   You must implement the design before generating a bitstream to configure your Xilinx device. During implementation, the Vivado tools place design elements onto device resources, route the design network, and optimize to reduce power and close timing. For more information, see the *Vivado Design Suite User Guide: Synthesis* (UG901) and *Vivado Design Suite User Guide: Implementation* (UG904).

8. Generate PCB Data (XDC, CSV, IBIS)

   I/O and clock planning is an iterative process that includes the exchange of information between the PCB or system designer and the FPGA designer. It can begin with input from the PCB using a target device pinout imported from a CSV file. When you complete the steps in the I/O and clock planning flow, you can return the pinout, along with device models for signal integrity analysis, using a comma-separated values (CSV) file and I/O buffer information specification (IBIS) models. For more information, see Interfacing with the System Designer.

Related Information

Defining Alternate Compatible Parts
I/O and Clock Planning Using the Platform Board Flow
Setting Device Configuration Modes
Setting Device Constraints
Configuring I/O Ports
Clock Planning
I/O Planning for UltraScale Architecture Memory IP
I/O and Clock Planning for IP with I/O Ports
Placing I/O Ports
Automatically Placing I/O Ports
Running DRCs
Working with SSN Analysis
Interfacing with the System Designer

I/O and Clock Planning Features

The following table shows the features supported for each type of project.

Table 2: **I/O and Clock Planning Features**

Feature	I/O Planning Project	RTL Design	Synthesized Design	Implemented Design
Read Ports from CSV and XDC Files	Supported[1]	N/A	N/A	N/A
Create or Delete Ports	Supported[1]	N/A	N/A	N/A
Read I/O Standard and Placement from XDC Files	Supported[1]	Supported[1]	Supported[1]	Supported[1]
Set Part Compatibility	Supported[1]	Supported	Supported	Supported
Set Configuration Mode	Supported[1]	Supported	Supported	Supported
I/O Planning DRCs	Supported[1]	Supported	Supported	Supported
Simultaneous Switching Noise (SSN) Analysis	Supported	Supported	Supported	Supported
Clock-Aware Placement and Clock-Aware DRC	N/A	N/A	Supported	Supported
Final Sign-Off DRC	N/A	N/A	N/A	Supported

Notes:

1. The Zynq® UltraScale+™ MPSoC multiplexed I/O (MIO) pins are defined when configuring the Zynq UltraScale+ MPSoC IP. These pins are only visible in the IP configuration, the implementation IO report, or the CSV generated from an elaborated, synthesized, or implemented design.

I/O and Clock Planning Using the Platform Board Flow

In the Vivado Design Suite, you can select pre-configured targeted design platform boards as the target for a design. Information about each platform board, including the target Xilinx device or devices, additional board components, signal interfaces, I/O configurations, and various preferred IP configuration options, is stored in a Board Interface file. The Vivado Design Suite provides a set of Board Interface files for predefined boards, and you can also define your own targeted platform boards for use in the Vivado tools. For more information on the platform board flow, see this link in the *Vivado Design Suite User Guide: System-Level Design Entry* (UG895).

I/O requirements can change based on the IP configurations and signal interfaces used when you customize an IP. When working with IP from the Vivado IP catalog in the platform board flow, you can automatically define the package pin assignment and all of the I/O-related constraints, such as IOSTANDARD, SLEW, and DRIVE.

In addition, various Vivado Design Suite Tcl commands let you access information in the Board Interface file while working in an I/O planning project, an RTL design project, or a post-synthesis netlist project. You can use the information from the Board Interface file to group ports and define interfaces or to define the required ports for a specific FPGA configuration. For more information, see Automatically Inferring I/O Port Interfaces.

Related Information

Automatically Inferring I/O Port Interfaces

I/O and Clock Planning for SSI Technology Devices

I/O and clock planning are critical when working with stacked silicon interconnect (SSI) technology. Because SSI technology devices have a larger die size, poor placement can create longer routes that increase power consumption and reduce performance. For information on pinout selection and clocking, see this link in the *UltraFast Design Methodology Guide for Xilinx FPGAs and SoCs* (UG949).

I/O and Clock Planning for IP with I/O Ports

Certain types of IP, such as Memory, GT, PCIe, and Ethernet interfaces have I/O ports associated with them. You must properly configure this IP using the IP capabilities in the Vivado Design Suite prior to beginning the I/O planning process. Because these interfaces are usually the most timing critical, use this IP as the starting point when considering the device pin assignments. In addition, use an RTL or synthesized design for the I/O pin planning process when using this IP.

Define the I/O physical pin assignments for the GT, PCIe IP, Ethernet, and Memory IP as part of IP customization when the core is added to the design. To change the I/O assignments, re-customize the IP in the design. For information on working with and customizing IP, see the *Vivado Design Suite User Guide: Designing with IP* (UG896). For UltraScale architecture Memory IP, the I/O assignment is integrated into the standard I/O planning flow and does not require Memory IP customization. For more information, see I/O Planning for UltraScale Architecture Memory IP.

Related Information

I/O Planning for UltraScale Architecture Memory IP

I/O Planning for Zynq UltraScale+ MPSoCs

Because of the advanced capabilities of the Zynq® UltraScale+™ MPSoCs, the pin planning flow differs from other devices. You must go through IP customization for the Zynq UltraScale+ MPSoC IP to indicate the features you plan to use in your design. As you go through the design flow, the multiplexed I/O (MIO) pins do not show up in the user design or constraints. The I/O Planning Project does not show the utilization of the MIO ports, nor does it write out the placement for the MIO pins. The only way to see a full list of the used pins for a schematic or for communicating with the board designer, is by using the **File → Export → Export I/O Ports** command to generate a CSV file. Xilinx recommends using an HDL project for all pin planning in Zynq UltraScale+ MPSoCs. For more information, see Defining and Configuring I/O Ports.

Related Information

Defining and Configuring I/O Ports

I/O Planning for UltraScale and UltraScale+

The bank types names are different in UltraScale and UltraScale+ devices from what they are in 7 series devices. Additional I/O standards have also been added. In 7 series devices, the bank types were High Range and High Performance; in UltraScale architecture devices, the names are High Density (replacing High Range), and High Performance. For more information, see this link in the *UltraScale Architecture SelectIO Resources User Guide* (UG571).

I/O Planning for Versal ACAP

The two primary types of I/O in Versal ACAPs are high-performance XP I/O (XPIO) and high density HD I/O (HDIO). The XPIO includes dedicated logic to support high-speed interfaces with voltage ranges between 1.0V and 1.5V. HDIO and XPIO banks do not have overlapping voltages or I/O standards. The HDIO supports interfaces with voltages ranging from 1.8V to 3.3V. The HDIO provides logic for both single data rate (SDR) and double data rate (DDR) interfaces at reduced clocking speeds. See *Versal ACAP SelectIO Resources Architecture Manual* (AM010) for architecture information and *Advanced I/O Wizard LogiCORE IP Product Guide* (PG320) for detailed information on Advanced IO Wizard / Advanced IO Planner.

All Versal devices have configurable SelectIO interface drivers and receivers, supporting a wide variety of standard interfaces. The robust feature set includes programmable control of output strength and slew rate, on-chip termination, and an internally generate a reference voltage (INTERNAL_VREF). Each Versal device contain XPIO banks that contain 54 SelectIO pins and can implement both single-ended and differential I/O standards. XPIO banks support the highest speed interfaces powered at or below 1.5V. Some Versal devices contain HDIO banks that can interface with voltage levels between 1.8V and 3.3V. The HDIO banks contain 22 SelectIO pins that can implement both single-ended I/O standards and differential I/O standards. Every SelectIO IOB resource contains input, output, and tristate drivers. The SelectIO pins can be configured to various I/O standards, both single-ended and differential.

- Single-ended I/O standards are, for example, LVCMOS, LVTTL, HSTL, SSTL, HSUL, LVSTL, and POD.
- Pseudo-differential standards are, for example, differential HSTL, POD, HSUL, LVSTL, and SSTL.
- LVDS compatible.

Chapter 2

Configuring the Device

The Vivado Design Suite provides several device planning features that can affect I/O and clock planning. For example, the ways in which you plan to configure the device, set device constraints, and set configuration voltage all have an impact on I/O and clock planning.

Optionally, you can also define alternate package-compatible devices to enable changing the size of your target part in case your final design requires it. Define these device-specific properties *before* beginning I/O and clock planning.

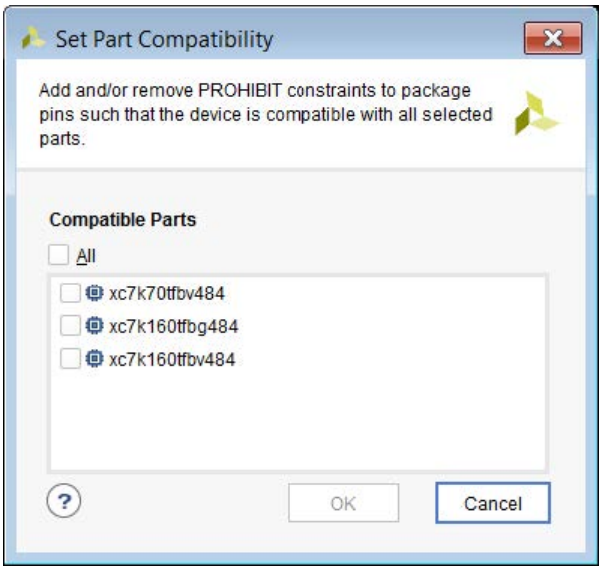
Defining Alternate Compatible Parts

You can select compatible devices for the design to allow you to retarget the design to alternate Xilinx® devices if necessary. The Vivado tools select compatible Xilinx devices in the same package as the current target part to preserve as much of the I/O assignment as possible. This ensures that the I/O pin assignments work in the selected alternate devices.

To define an alternate compatible part:

1. Select **Tools → I/O Planning → Set Part Compatibility**.
2. In the Set Part Compatibility dialog box, select the alternate parts, and click **OK**.

Figure 2: **Set Part Compatibility Dialog Box**



The Vivado IDE identifies the pins that are common to all selected alternate parts, and assigns PROHIBIT constraints to pins that are not common to all devices. The number of pins available for placement might be reduced when you select additional alternate parts.

In addition, the Vivado IDE automatically prohibits signals from being assigned to any unbonded pins in the selected alternate devices. A dialog box displays the number of prohibited package pins. You can view prohibits in the Package, Package Pins, and Device windows. Prohibited pins are indicated by a slashed circle icon .

Tcl Command Example for Defining Alternate Compatible Parts

```
set_property KEEP_COMPATIBLE xc7k160tfbg676 [current_design]
```

Note: For more information on the set_property Tcl command and other Tcl commands, see the *Vivado Design Suite Tcl Command Reference Guide* (UG835). For more information on the KEEP_COMPATIBLE property and other properties, see the *Vivado Design Suite Properties Reference Guide* (UG912).

Setting Device Configuration Modes

Because Xilinx device configuration data is stored in CMOS latches, the device must be reconfigured each time it is powered up. The bitstream is loaded into the device through special mode configuration pins that serve as the interface for a number of different configuration modes. The specific configuration mode is selected by setting the appropriate voltage level on dedicated input pins.

Each configuration mode has a corresponding set of interface pins that span one or more I/O banks on the device. Bank 0 contains the dedicated configuration pins and is always part of every configuration interface. Bank 65 in UltraScale and UltraScale+ devices and Banks 14 and 15 in 7 series devices contain multifunction pins that are used in various configuration modes. For information on the available device configuration modes, see the Configuration User Guide for your device. For information on analyzing how the configuration modes might conflict with other multifunction pins, see Multifunction Pins.

To set device configuration modes and view information about the modes:

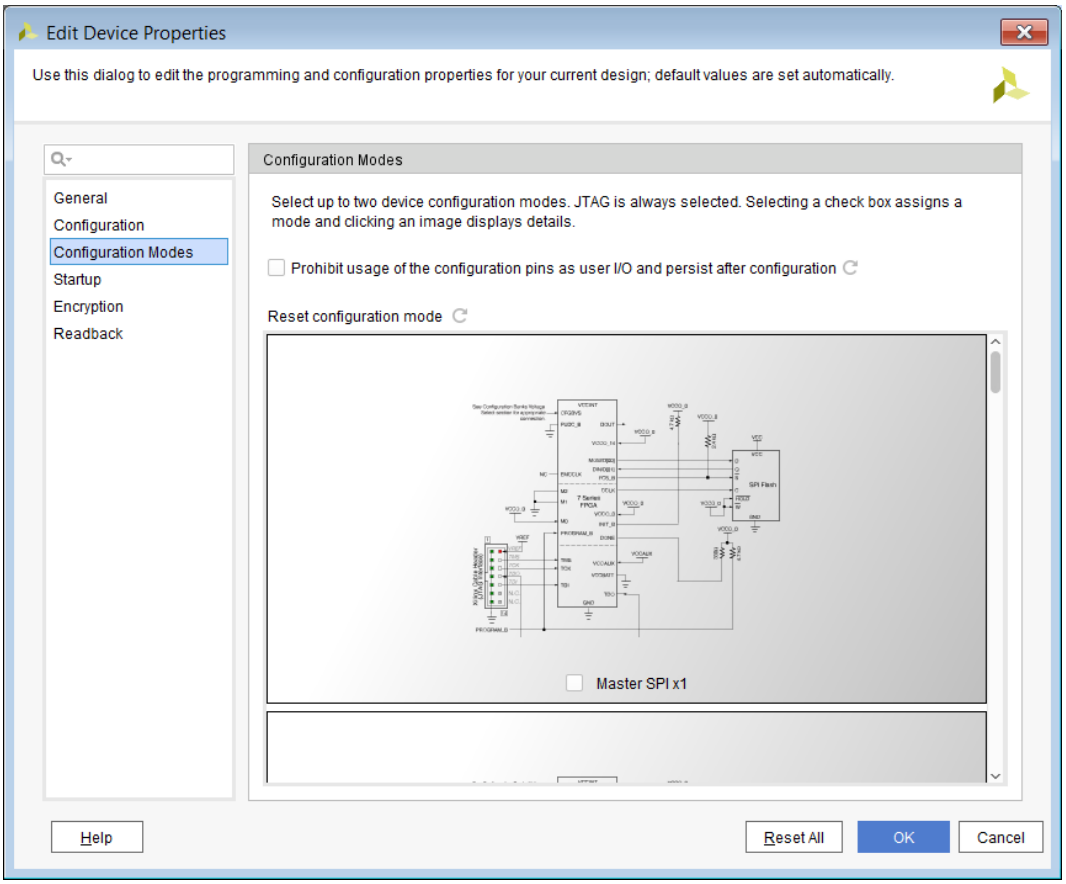
1. Select **Tools → Edit Device Properties**.
2. In the Edit Device Properties dialog box (see the following figure), select the Configuration Modes category, do the following, and click **OK** to close the dialog box:
   - Enable the check box for a configuration mode to set that configuration mode. When you set a configuration mode:
     - The associated I/O pins display in the Config column of the Package Pins window.
     - The following constraints are created when you save the design:

       ```
       set_property BITSTREAM.CONFIG.PERSIST NO [current_design]
       set_property CONFIG_MODE <configuration_mode> [current_design]
       ```
   - Click a configuration mode to open a dialog box in which you can view information, including a description, configuration diagram, and links to more information. Click **Print** to print the configuration diagram.
   - Enable **Prohibit usage of the configuration pins as user I/O and persist after configuration** to ensure that pins are used as configuration pins and not as general purpose I/Os after configuration. When you select this option, the following constraint is created when you save the design:

     ```
     set_property BITSTREAM.CONFIG.PERSIST YES [current_design]
     ```

     Note: For more information on the configuration bitstream settings, see this link in the *Vivado Design Suite User Guide: Programming and Debugging* (UG908).

IMPORTANT! *The JTAG configuration mode is always selected. You can select one configuration mode in addition to the JTAG configuration mode.*



3. Select **File → Constraints → Save** to save the constraints to the target XDC file.

   Note: The Tcl command equivalent is: `save_constraints`

> **TIP:** *When setting device configuration modes, you can undo your last action using* ***Edit → Undo****. Alternatively, you can enter* `undo` *in the Tcl console.*

Related Information

Multifunction Pins

Tcl Command Example for Setting Device Configuration Modes

```
set_property CONFIG_MODE SPIx2 [current_design]
```

Note: By default, configuration pins are not set to persist after configuration. To ensure that pins are used as configuration pins and not as general purpose I/Os after configuration, enter the following Tcl command:

```
set_property BITSTREAM.CONFIG.PERSIST YES [current_design]
```

Setting Device Constraints

In the Device Constraints window (shown in the following figure), you can set constraints, including DCI_ CASCADE and INTERNAL_VREF. Xilinx devices have configurable SelectIO™ interface drivers and receivers that support many standard interfaces. This feature set includes programmable control of output strength and slew rate, on-chip termination using digitally-controlled impedance (DCI), and the ability to internally generate a reference voltage (INTERNAL_VREF).

Depending on the I/O standard, Xilinx DCI can either control the output impedance of a driver or add a parallel termination to the driver, receiver, or both, with the goal of accurately matching the characteristic impedance of a transmission line. DCI actively adjusts the impedance inside an I/O bank to calibrate to external precision reference resistors placed on the VRN and VRP pins. This compensates for changes in I/O impedance due to process variation or variations of temperature and supply voltage. DCI uses two multi-purpose reference pins in each I/O bank to control the impedance of the driver or the parallel-termination value for all of the I/Os in the bank.

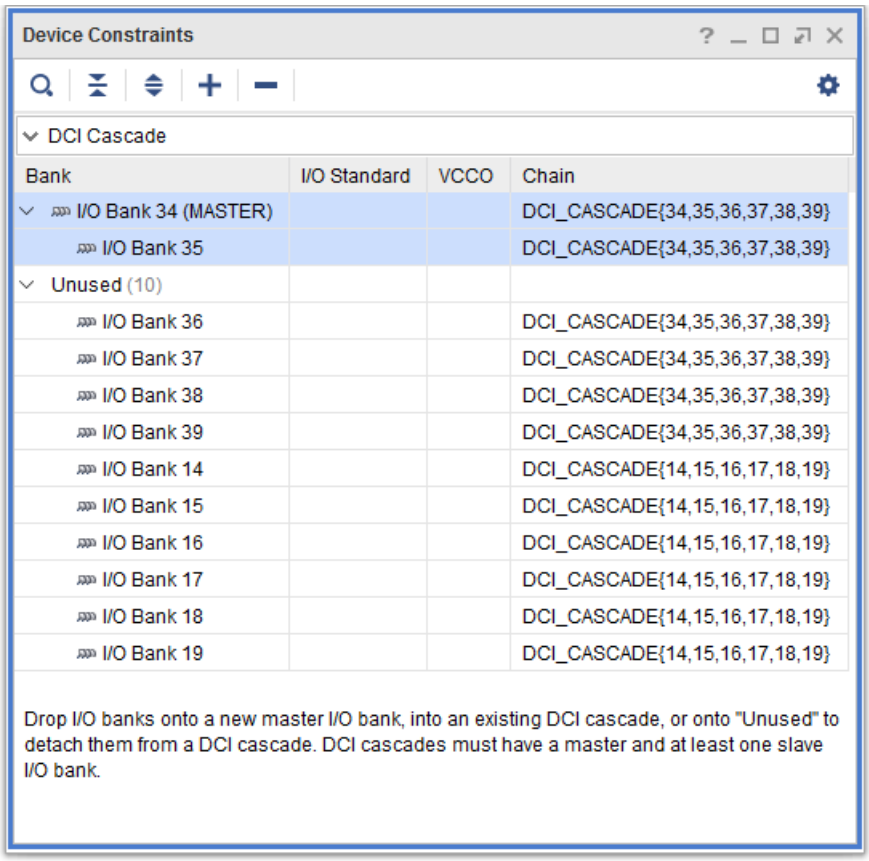
Single-ended I/O standards with a differential input buffer require a VREF. When a VREF is required within an I/O bank, use the following pins for the bank as VREF supply inputs:

- For UltraScale architecture-based devices, use the dedicated VREF pin
- For 7 series devices, use the two multifunction VREF pins

Alternatively, you can generate an internal VREF using the INTERNAL_VREF constraint. Using internal reference voltages can remove the need to provide a particular VREF supply rail on the PCB and can free multipurpose VREF pins in a given I/O bank for other I/O port assignments. Each I/O bank has one VREF plane, and each bank can have the optional INTERNAL_VREF set to a single voltage level for the entire bank.

For more information, see the *7 Series FPGAs SelectIO Resources User Guide* (UG471) or the *UltraScale Architecture SelectIO Resources User Guide* (UG571) depending on your device.

Figure 3: **Device Constraints Window**

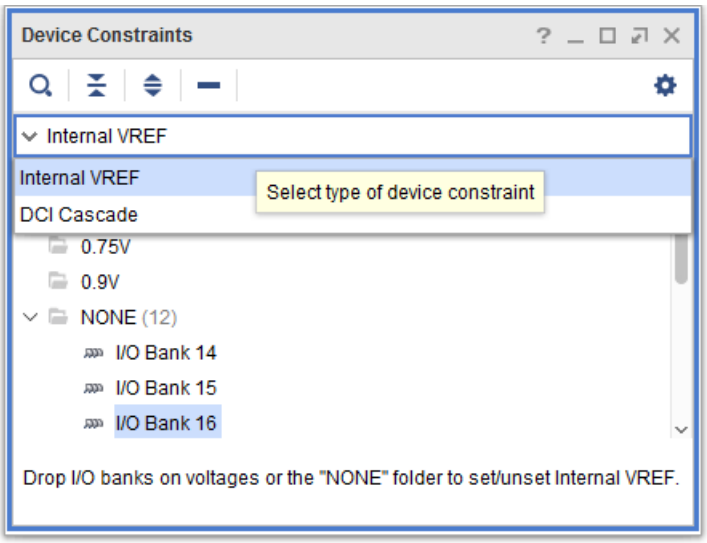


Creating a DCI_CASCADE Constraint

The DCI_CASCADE constraint links two or more adjacent I/O banks together for DCI reference voltage purposes. The I/O bank with the DCI reference voltage is the master, and all other I/O banks in the cascade are slaves. All banks in a cascade must be in the same I/O column of the device.

Note: You can configure the DCI_CASCADE constraint for 7 series devices, Zynq-7000, and UltraScale architecture-based devices. For more information on this constraint, see DCI_CASCADE in the *Vivado Design Suite Properties Reference Guide* (UG912). To create a DCI_CASCADE constraint:

1. In the Device Constraints window, select **DCI Cascade** from the drop-down list at the top of the window.



2. Select the I/O banks you want to cascade, right-click, and select **Add DCI Cascade**.
3. In the Add DCI Cascade dialog box, select an I/O bank to be the master of the new DCI cascade, and click **OK**.

   The master bank appears in the Device Constraints window.

   Note: DCI cascades must have a master and at least one slave I/O bank.

> **TIP:** *Alternatively, you can create a DCI_CASCADE constraint in the Package window or Package Pins window. Right-click the banks to cascade, and select* ***Create a DCI Cascade****.*

Tcl Command Example for Creating a DCI_CASCADE Constraint

```
set_property DCI_CASCADE {31 32} [get_iobanks 36]
```

Note: When using this Tcl command, `get_iobanks` specifies the master bank. In this example, 31 and 32 are the slave banks, and 36 is the master bank.

Modifying or Removing DCI Cascade Constraints

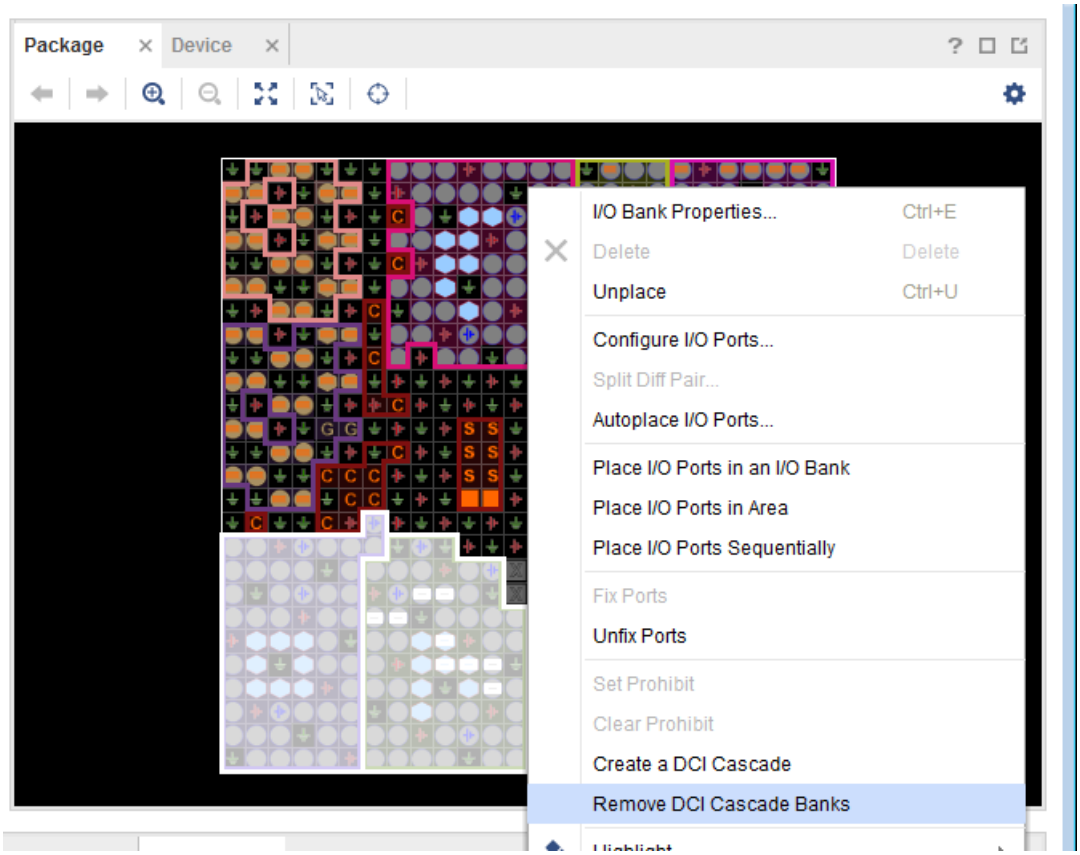
To modify DCI cascades, do any of the following in the Device Constraints window:

- To change the master, right-click a DCI cascade, and select **Add DCI Cascade**. In the Add DCI Cascade dialog box, select a different bank to be the master.
- To add an I/O bank to a DCI cascade, drag and drop the I/O bank onto the DCI cascade.
- To remove an I/O bank from a DCI cascade, drag and drop the I/O bank onto the Unused folder.
- To remove an entire DCI cascade, right-click the DCI cascade, and select **Remove DCI Cascade Banks**).

Note: The Tcl command equivalent for this action is:

```
set_property DCI_CASCADE {} [get_iobanks 36]
```

Figure 4: **Removing DCI Cascade Banks**



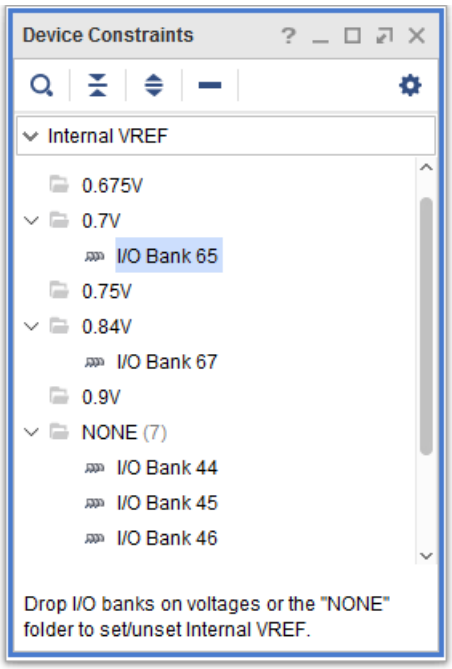
Creating an INTERNAL_VREF Constraint

Xilinx devices can optionally use an internally generated reference voltage by enabling the INTERNAL_VREF constraint. Internal generation removes the need to provide for a particular VREF supply rail on the PCB and frees the multipurpose VREF pins in a given I/O bank to be used as normal I/O pins.

TIP: *All I/O banks that do not have an INTERNAL_VREF constraint are shown under the NONE folder in the Device Constraints window.*

To create an INTERNAL_VREF constraint, drag and drop the I/O bank onto the desired voltage folder (for example, 0.7V or 0.84V) in the Device Constraints window.

Figure 5: **Creating an Internal V_{REF} Constraint**



Tcl Command Example for Creating an INTERNAL_VREF Constraint

```
set_property INTERNAL_VREF 0.7 [get_iobanks 65]
set_property INTERNAL_VREF 0.84 [get_iobanks 67]
```

Setting the Configuration Bank Voltage Select Pin

The configuration bank voltage select (CFGBVS) logic input pin is referenced between VCCO_0 and GND. The CFGBVS pin must be set to High or Low to determine the I/O voltage support for the pins in bank 0. In the Vivado tools, you can use Tcl commands to set the CFGBVS tie off information to either VCCO or GND. You can set the configuration voltage, or VCCO_0 voltage, to 1.5, 1.8, 2.5, or, 3.3. Based on these settings, DRCs are run on Bank 0, 14, and 15 for 7 series devices. For UltraScale devices, DRCs are run on Bank 0 and 65. These values are also used when exporting IBIS models.

Following is an example:

```
set_property CFGBVS VCCO [current_design]
set_property CONFIG_VOLTAGE 3.3 [current_design]
```

By default, the CFGBVS property is empty. The Vivado tools check whether the CFGBVS property is set to VCCO or GND. If the CFGBVS property has a value, the Vivado tools check for a CONFIG_MODE property. DRCs are issued based on IOSTANDARD and CONFIG_VOLTAGE settings for the bank.

When you export to a CSV file, the Vivado tools provide VCCO tie off information for the relevant banks (for 7 series devices: banks 0, 14, and 15; for UltraScale architecture-based devices: banks 0 and 65) based on the setting for the CONFIG_MODE property. For example, if you use JTAG/Boundary Scan, CFGBVS is GND, and CONFIG_VOLTAGE is 3.3, the tools issue the critical warning: DRC CFGBVS-4. This indicates that the CONFIG_VOLTAGE is set to and must instead be set to VCCO, which has a value of 1.8. For UltraScale+ devices, you cannot manually set CFGBVS or CONFIG_VOLTAGE. By default, CFGBVS is set to GND and CONFIG_VOLTAGE is set to 1.8V.

***Note*:** For more information on the CFGBVS pin, see one of the following, depending on your device:

- *7 Series FPGAs Configuration User Guide* (UG470)
- *UltraScale Architecture Configuration User Guide* (UG570)
- *Zynq UltraScale+ Device Technical Reference Manual* (UG1085)

Chapter 3

I/O Logic and Low-Speed I/O Planning

You can use the I/O Planning view layout in the Vivado IDE on elaborated, synthesized, and implemented designs. The view layout uses both Device and Package windows. Additional I/O information appears in the following windows: Clock Resources, Clock Regions, Package Pins, I/O Ports, Device Constraints, and Properties windows.

The Advanced I/O Planner is a new tool for Versal devices that allows you to place and move around I/O on a nibble and bank level granularity. Low speed I/O planning is done through the traditional design methodology as in I/O planning. For additional details on pin planning low speed I/O see *Versal ACAP SelectIO Resources Architecture Manual* (AM010).

***Note*:** For more information on the windows in the Vivado IDE, see the *Vivado Design Suite User Guide: Using the Vivado IDE* (UG893).

TIP: *When working with I/O planning projects, the I/O Planning view layout is called the Default Layout.*

The I/O Planning view layout provides an interface to:

- Create, import, and configure the initial list of I/O ports early in the design flow.
- Perform final verification of the pinout at the end of the design flow.
- Group related ports into interfaces, then assign them to package pins.
- Use fully automatic pin placement or semi-automatic interactive modes for controlled I/O port assignment.
- View the relationship of the physical package pins and banks with their corresponding I/O die pads.
- Make informed decisions to optimize the connectivity between the PCB and the Xilinx®device.
- Analyze the design and device I/O requirements.
- Define an I/O pinout configuration or pinout that satisfies the requirements of both PCB and FPGA designs.

Using the I/O Planning View Layout

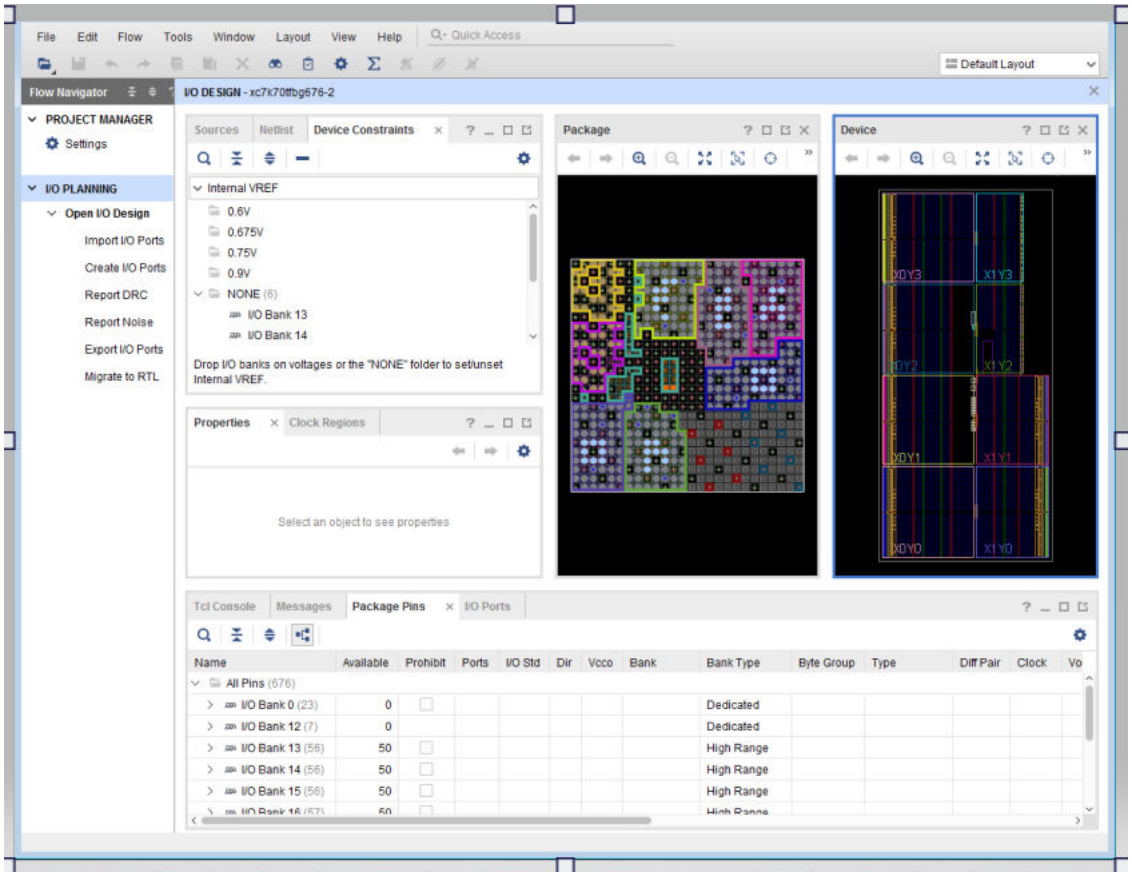
To launch the I/O Planning view layout, open an elaborated, synthesized, or implemented design, and do any of the following:

- Select **Layout→I/O Planning**.
- From the Layout selector, select **I/O Planning**.
- Using the New Project wizard, create a new I/O Planning Project.

  Note: For more information on creating an I/O planning project, see this link in the *Vivado Design Suite User Guide: System-Level Design Entry* (UG895).

The following figure shows the I/O Planning view layout.

Figure 6: **I/O Planning View Layout**



Viewing Device Resources

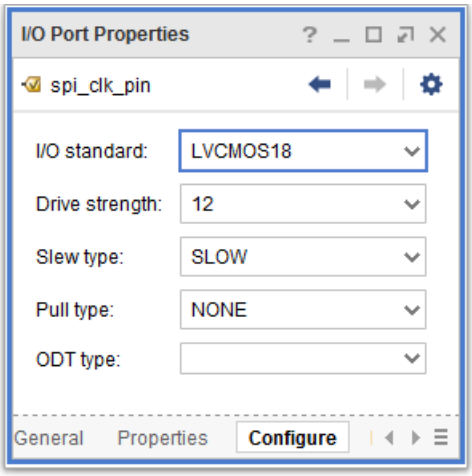
The Device and Package windows show a graphical representation of the device and placed logic resources. When a logic object or device site is selected in these windows, information appears in the Properties window. The following sections describe these windows in detail.

TIP: *To search for specific objects or device sites, use the* ***Edit → Find*** *command. The Find dialog box includes many searchable object types and robust filtering capabilities to search the device or design for specific objects. You can select objects directly from the Find Results window.*

Properties

The Properties window shows properties for a selected object. The title bar changes to reflect the type of object selected. In most cases, the Properties window includes various views to show different information about the object. For example, in the following figure, the Properties window shows the I/O Port Properties and includes the General, Properties, and Configure views. To show the Properties window, select **Window → Properties**.

Figure 7: **I/O Port Properties**



TIP: *You can get property information on package pins using Tcl commands. For example, the following command shows all the properties associated with the specified package pin:*

```
report_property [get_package_pins <pin_number>]
```

The following command shows the maximum trace delay for the specified package pin:

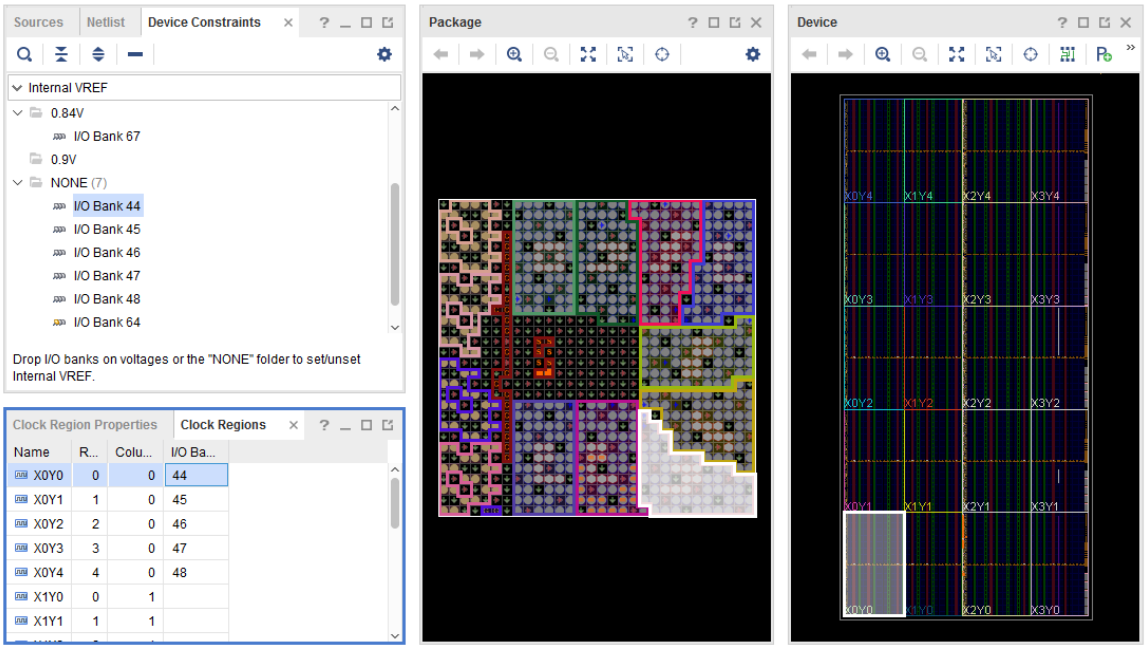
```
get_property MAX_DELAY [get_package_pins <pin_number>]
```

For more information on Tcl commands, see the Vivado Design Suite Tcl Command Reference Guide (UG835).

Clock Region Resources and Statistics

The Clock Regions window allows easy selection of the clock regions. When you select a clock region in the Clock Regions window, the related I/O banks and regional clock resources are highlighted in the Package and Device windows, as shown in the following figure.

Figure 8: **Clock Regions Window**

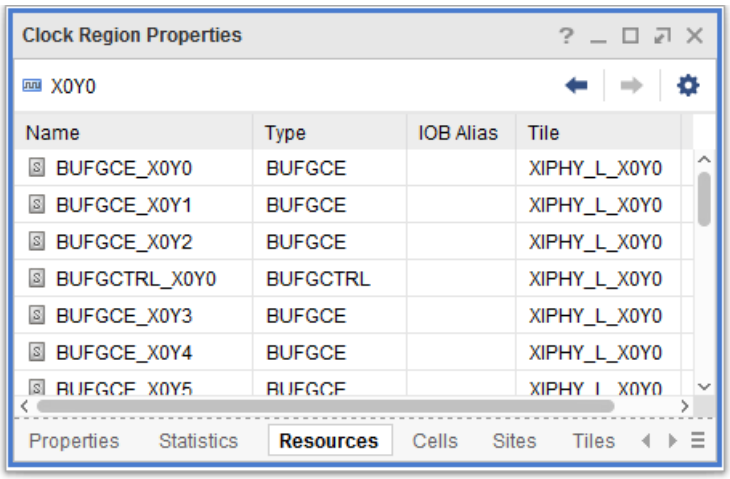


When clock regions are highlighted, you can click the **Clock Region Properties** tab to view the properties for the selected clock region. In the Clock Region Properties window, you can:

- Select the **Statistics** view to display the resource statistics available within the clock region as well as the logic content of the selected clock region.
- Select the **Resources** view to locate device clock resources for logic assignment, as shown in the following figure.

***Note*:** When you select an object in the Clock Regions Properties window, the object is cross-selected in another open window, such as the Device window. By tiling the windows it is possible to see both the Device and Package windows simultaneously.

Figure 9: **Clock Region Properties—Resources View**



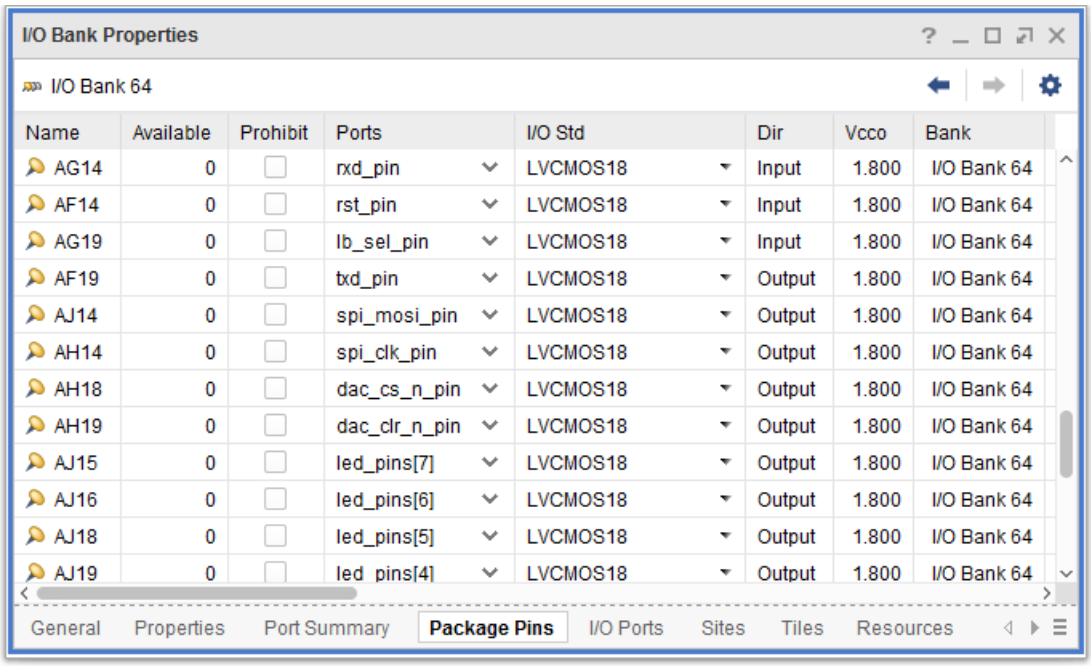
I/O Bank Resources

You can select I/O resources in any of the I/O planning windows and their corresponding data is highlighted in all other windows as shown in the Clock Regions Window figure in Clock Region Resources and Statistics. This provides a visual indication of the relationship between the physical package and the internal die.

To access information about a specific I/O bank:

1. In the Package Pins window, select an I/O bank.
2. In the I/O Bank Properties window, click the views at the bottom of the window to see the different types of information available.

Figure 10: **I/O Bank Properties**



Name	Available	Prohibit	Ports	I/O Std	Dir	Vcco	Bank
AG14	0		rxd_pin	LVCMOS18	Input	1.800	I/O Bank 64
AF14	0		rst_pin	LVCMOS18	Input	1.800	I/O Bank 64
AG19	0		lb_sel_pin	LVCMOS18	Input	1.800	I/O Bank 64
AF19	0		txd_pin	LVCMOS18	Output	1.800	I/O Bank 64
AJ14	0		spi_mosi_pin	LVCMOS18	Output	1.800	I/O Bank 64
AH14	0		spi_clk_pin	LVCMOS18	Output	1.800	I/O Bank 64
AH18	0		dac_cs_n_pin	LVCMOS18	Output	1.800	I/O Bank 64
AH19	0		dac_clr_n_pin	LVCMOS18	Output	1.800	I/O Bank 64
AJ15	0		led_pins[7]	LVCMOS18	Output	1.800	I/O Bank 64
AJ16	0		led_pins[6]	LVCMOS18	Output	1.800	I/O Bank 64
AJ18	0		led_pins[5]	LVCMOS18	Output	1.800	I/O Bank 64
AJ19	0		led_pins[4]	LVCMOS18	Output	1.800	I/O Bank 64

Related Information

Clock Region Resources and Statistics

Multifunction Pins

The Package Pins window contains several data types that display in columns similar to a spreadsheet. In this window, you can:

- Flatten, filter, and sort data.
- Move, hide, and configure columns to display and compare the various multifunction pins.
- Directly edit certain cells by entering text or selecting a value from drop-down menus.

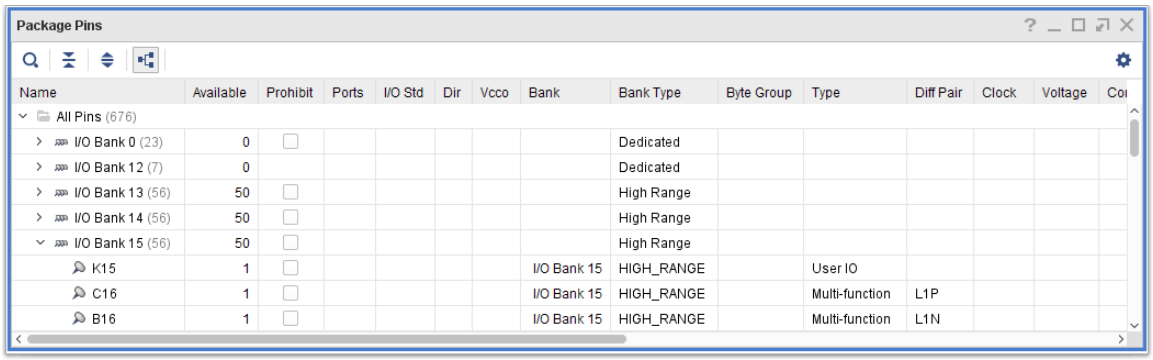
The Package Pins window includes the following information:

- Type column identifies multifunction pin types.
- Config column shows the pin definition of the multifunction pin after you set the device configuration mode.

  Note: Many device configuration modes use multifunction pins. For more information, see Setting Device Configuration Modes.
- Other columns contain information about logic or configuration modes involving multifunction type pins.

- Information identifies conflicting multifunction pins for designs that contain GTs, memory controllers, or PCI™ logic.
- Asterisks indicate the I/O standard or attribute, such as drive strength or slew type, is not set to the default generated by the system.

Figure 11: **Package Pins Window**



In the Package window, the following symbols indicate the function of multifunction pins:

- Clock capable pins display as a hexagon icon
- V_{REF} pins display as a small power icon

  For Zynq UltraScale+ MPSoCs, pins controlled by the configuration of the Zynq UltraScale+ MPSoC IP display as a square icon .

> **IMPORTANT!** *Dedicated I/O pins are dedicated to the targeted device not to the bank. For example, dedicated I/O pins such as VCCO and GND are device-specific rather than bank-specific.*

Related Information

Setting Device Configuration Modes

Defining and Configuring I/O Ports

You can use the Vivado IDE to import, create, and configure I/O ports as described in the following sections.

Importing I/O Ports

Depending on the project type, you can use the following methods to import I/O ports:

- **I/O Planning Project:** You can import XDC and CSV files into an empty I/O planning project when you create the project or later using the file import capability. For details, see Importing a CSV File and Importing an XDC File.
- **RTL Project:** Use RTL files or headers to create an RTL project for I/O planning, then add more complete RTL source files to the project later as the design progresses. When you create an RTL-based or synthesized netlist-based project, the I/O Ports window automatically populates with the I/O ports defined in the design.
- **Migrate from I/O Planning Project to RTL Project:** You can convert an I/O planning project to an RTL project, turning the I/O ports into a top-level Verilog or VHDL module definition for the design. For more information, see Migrating an I/O Planning Project to an RTL Project.

Related Information

Importing a CSV File
Importing an XDC File
Migrating an I/O Planning Project to an RTL Project

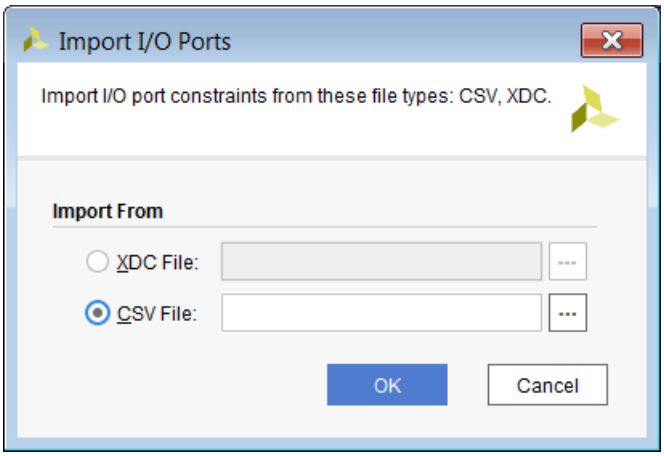
Importing a CSV File

You can import a CSV file to populate the I/O Ports window within the I/O Planning layout view. You can then assign these I/O ports to physical package pins to define the device pin configuration.

To import an I/O ports list from a CSV file:

1. Select **File → Import → Import I/O Ports**.
2. In the Import I/O Ports dialog box, select **CSV File**, and browse to select the file to import.

Figure 12: **Import I/O Ports Dialog Box**



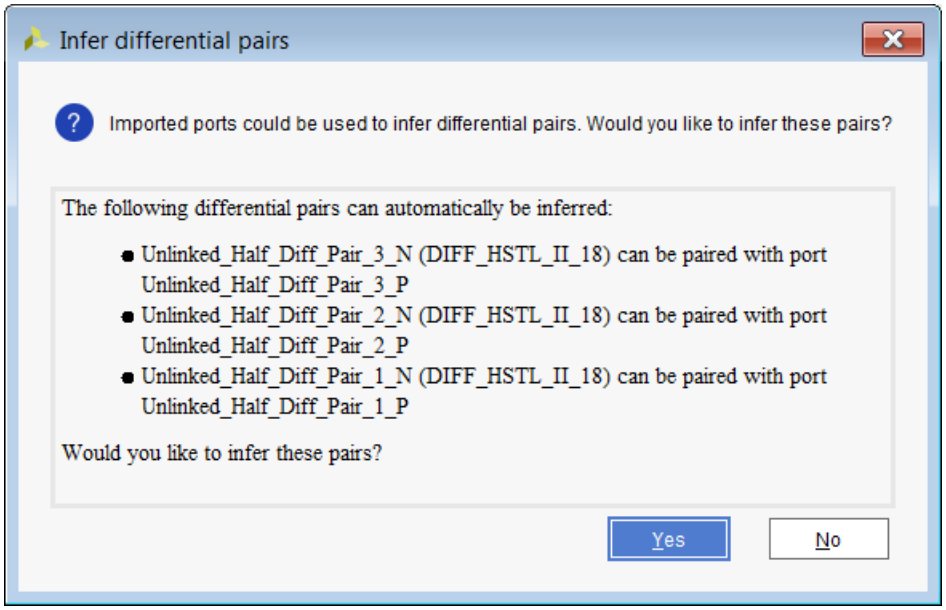
The following figure shows the CSV file format. CSV is a standard file format used by FPGA and board designers to exchange information about device pins and pinout. The Vivado IDE requires a specific CSV file format for importing I/O pin-related data, as described in Using I/O Port Lists in CSV File Format.

Figure 13: **I/O Port List in CSV File Format**

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
#Top: ios	Floorplan: io_1	Part: xc7k70tfbg484-2												
#Package Version: FINAL 2012-06-26														
#Package Pin Delay Version: VERS. 2.0 2012-06-26														
IO Bank	Pin Number	Site	Site Type	Min Trace	Max Trac	Prohibit	Interface	Signal Name	Direction	DiffPair Type	DiffPair Signal	IO Standard	Drive (mA)	Slew Rate
13	AB17	IOB_X0Y15	IO_L17N_T2_13	100.47	101.48			RXN_IN[7]	IN	N	RXP_IN[7]	DIFF_HSTL_II_18		
13	AA15	IOB_X0Y13	IO_L18N_T2_13	96.185	97.151			RXN_IN[6]	IN	N	RXP_IN[6]	DIFF_HSTL_II_18		
13	V17	IOB_X0Y11	IO_L19N_T3_VREF_13	64.754	65.405			RXN_IN[5]	IN	N	RXP_IN[5]	DIFF_HSTL_II_18		
13	T16	IOB_X0Y9	IO_L20N_T3_13	57.59	58.169			RXN_IN[4]	IN	N	RXP_IN[4]	DIFF_HSTL_II_18		
13	Y16	IOB_X0Y7	IO_L21N_T3_DQS_13	74.609	75.359			RXN_IN[3]	IN	N	RXP_IN[3]	DIFF_HSTL_II_18		
13	Y14	IOB_X0Y5	IO_L22N_T3_13	94.887	95.841			RXN_IN[2]	IN	N	RXP_IN[2]	DIFF_HSTL_II_18		
13	W15	IOB_X0Y3	IO_L23N_T3_13	64.191	64.837			RXN_IN[1]	IN	N	RXP_IN[1]	DIFF_HSTL_II_18		
13	U15	IOB_X0Y1	IO_L24N_T3_13	57.777	58.358			RXN_IN[0]	IN	N	RXP_IN[0]	DIFF_HSTL_II_18		
13	AA16	IOB_X0Y16	IO_L17P_T2_13	96.864	97.837			RXP_IN[7]	IN	P	RXN_IN[7]	DIFF_HSTL_II_18		
13	AA14	IOB_X0Y14	IO_L18P_T2_13	108.261	109.349			RXP_IN[6]	IN	P	RXN_IN[6]	DIFF_HSTL_II_18		
13	U16	IOB_X0Y12	IO_L19P_T3_13	57.901	58.483			RXP_IN[5]	IN	P	RXN_IN[5]	DIFF_HSTL_II_18		
13	R16	IOB_X0Y10	IO_L20P_T3_13	61.145	61.76			RXP_IN[4]	IN	P	RXN_IN[4]	DIFF_HSTL_II_18		
13	W16	IOB_X0Y8	IO_L21P_T3_DQS_13	70.835	71.547			RXP_IN[3]	IN	P	RXN_IN[3]	DIFF_HSTL_II_18		
13	W14	IOB_X0Y6	IO_L22P_T3_13	109.229	110.327			RXP_IN[2]	IN	P	RXN_IN[2]	DIFF_HSTL_II_18		

You can define differential pairs in the CSV file in several ways. For example, the Vivado IDE recognizes differential pairs directly defined with DiffPair Signal and DiffPair Type properties. In addition, the Vivado IDE can infer diff pairs when only one port of the pair is defined in the CSV file or two named nets imply a differential pair. For more information, see Differential Pairs in the CSV File. When inferring differential pairs, the Vivado IDE displays a prompt to confirm the assignment of the pairs, as shown in the following figure.

Figure 14: **Infer Differential Pairs Dialog Box**



CSV files can also contain additional information not recognized by the Vivado IDE. If unrecognized information is found in the imported CSV file, the information is displayed in user columns of the Package Pins window for your review and use. To modify or define values in the user CSV fields, right-click in the Package Pins window, and select **Set User Column Values**.

Note: For information on exporting a CSV file, see Exporting I/O Pin and Package Data.

Related Information

Using I/O Port Lists in CSV File Format
Differential Pairs in the CSV File
Exporting I/O Pin and Package Data

Importing an XDC File

To import I/O port definitions from an XDC file:

1. Select **File → Import → Import I/O Ports**.
2. In the Import I/O Ports dialog box, select **XDC File**, and browse to select the file to import.

Because the XDC format does not define port direction, the direction is undefined. To define the I/O port direction, right-click in the I/O Ports window, and select **Set Direction**. You can also directly modify the direction of a specific I/O port in the I/O Ports window. For more information, see Setting I/O Port Direction.

Related Information

Setting I/O Port Direction

Creating Single-Ended or Differential I/O Ports

You can manually define new ports in an I/O planning project. Refer to Xilinx device documentation for information regarding voltage capabilities of the device.

Note: The I/O Ports window groups each differential pair into a single row. Because a single row represents two ports, the total number of ports shown in parentheses is higher than the number of rows. To get a list of signals that matches the total number of ports in the I/O Ports window, enter the following Tcl command:

```
get_ports * -filter {BUS_WIDTH == "" }
```

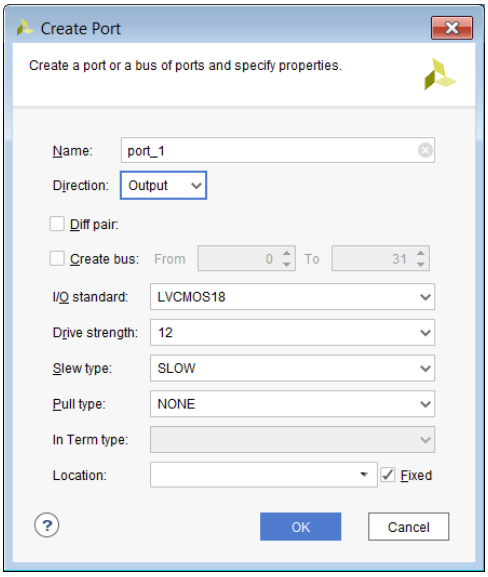
To create I/O ports:

1. In the I/O Ports window, right-click, and select **Create I/O Ports**.
2. In the Create I/O Ports dialog box, edit the following options, and click **OK**:
   - **Name:** Enter the port or bus name to create.

- **Direction:** Select the port direction.
- **Diff Pair:** Define differential pair signals or buses.

  Note: To create a differential I/O port, enable this option. This creates two ports and adds a `_N` suffix to the name of the negative port.
- **Create Bus:** Enter a bus range for bus creation.
- **I/O Standard:** Select the I/O standard constraint.
- **Drive Strength:** Select the drive strength value.
- **Slew Type:** Select the slew type value.
- **Pull Type:** Select the pull type value.
- **In Term Type:** Define the parallel termination properties of the input signal.

Figure 15: **Create I/O Ports Dialog Box**



Tcl Command Examples for Creating Single-Ended or Differential I/O Ports

- Creating a single-ended I/O port:

```
create_port port_1 -direction in
```

- Creating a differential I/O port:

```
create_port port_2 -direction in -diff_pair
```

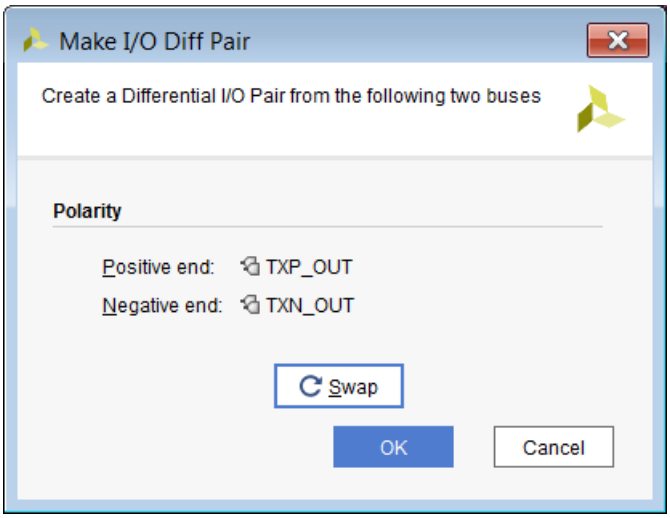
Making and Splitting Differential Pairs

To define a differential pin pair in an I/O planning project:

1. In the I/O Ports window, select any two I/O ports, right-click, and select **Make Diff Pair**.

   IMPORTANT! *The Make Diff Pair option is not available in RTL projects. In RTL projects, differential ports must be defined in the source code using appropriate I/O buffer instantiations.*

   In the Make I/O Diff Pair dialog box (see the following figure), the two I/O Ports display with Positive End and Negative End assignments made by the tool.



2. To reverse the Positive End and Negative End signals, click **Swap**, and click **OK**.

   TIP: *Right-click, and select **Split Diff Pair** to separate a diff pair into two ports.*

   IMPORTANT! *When you apply certain constraints to one side of a differential pin pair in an UltraScale™ architecture, the opposite constraint is automatically applied to the other side. For example, if you apply a PULLDOWN constraint to the P side, a PULLUP constraint is applied to the N side. If you make changes to the constraints, the most recent settings overwrite previous settings.*

Tcl Command Examples for Making and Splitting Differential Pairs

- Making a differential pair:

```
make_diff_pair_ports txp_out txn_out
```

- Splitting a differential pair:

```
split_diff_pair_ports txp_out txn_out
```

Configuring I/O Ports

Xilinx devices support configurable SelectIO™ interface drivers and receivers, which support various standard interfaces. These standard interfaces include programmable control of output strength and slew rate, on-chip termination using DCI, and generation of internal V_{REF}. You can configure one or more I/O ports to define I/O standard, drive strength, slew type, pull type, and in term. This is useful for configuring ports that were imported from CSV or XDC files without the appropriate characteristics. Configure these ports to support the standards required for the system-level design. For example, you can combine some I/O standards within a single I/O bank but not others.

For information on standards and requirements for I/O banks, see the following, depending on your device:

- *7 Series FPGAs SelectIO Resources User Guide* (UG471)
- *UltraScale Architecture SelectIO Resources User Guide* (UG571)

For information on packaging and pinout specifications, see the following, depending on your device:

- *7 Series FPGAs Packaging and Pinout Product Specification* (UG475)
- *UltraScale and UltraScale+ FPGAs Packaging and Pinouts Product Specification* (UG575)
- *Zynq-7000 SoC Packaging and Pinout Product Specifications* (UG865)

For detailed information on Zynq®-7000 pins, including MIO pins, see the *Zynq-7000 SoC Technical Reference Manual* (UG585).

For detailed information on Zynq UltraScale+ MPSoC pins, including MIO pins, see *Zynq UltraScale+ Device Technical Reference Manual* (UG1085).

To configure a port or a group of ports:

1. In the I/O Ports window, select the ports.
2. Right-click, and select **Configure I/O Ports**.
3. In the Configure Ports dialog box (see the following figure), edit the following options, and click **OK**.

   Note: The Configure Ports dialog box options vary depending on the targeted device.

   - **I/O standard:** Select the I/O standard constraint. The tool does not check the I/O standard when it is assigned. You can assign any I/O standard to any port, but this might result in errors when running DRCs.
   - **Drive strength:** Select the drive strength value.
   - **Slew type:** Select the slew type value.

- **Pull type:** Select the pull type value.
  - **PULLUP:** Applies a weak logic High level on a 3-stateable output or bidirectional port to prevent it from floating when not being driven.
  - **PULLDOWN:** Applies a weak logic Low level on a 3-stateable output or bidirectional port to prevent it from floating when not being driven.
  - **KEEPER:** Applies a weak driver on an 3-stateable output or bidirectional port to preserve its value when not being driven.
  - **NONE:** Does not apply a driver.

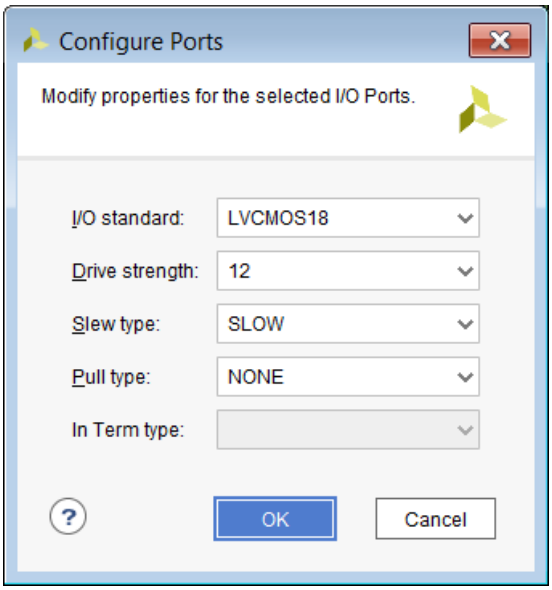
    ***Note*:** Alternatively, you can set the pull type constraint by clicking in the Pull Type column of the I/O Ports window.
- **In Term type:** (7 series devices only) Define the parallel termination properties of the input signal. For more information, see the *7 Series FPGAs SelectIO Resources User Guide* (UG471).
- **ODT:** (UltraScale architecture-based devices only) Define the value of the on-die termination (ODT) at the input for both DCI and non-DCI versions of the standards supported. For more information, see the *UltraScale Architecture SelectIO Resources User Guide* (UG571).
- **Fixed:** Indicates that the logical ports are user assigned. Ports must be fixed to ensure that the bitstream generates without errors.

In the Configure Ports dialog box, the Fixed option is read only. To fix ports, select the ports in the I/O Ports window, right-click, and select Fix Ports, or enter the following Tcl command in the Tcl Console:

```
set_property IS_LOC_FIXED true [get_selected_objects]
```

Alternatively, you can enter the following Tcl command to fix ports:

```
set_property IS_LOC_FIXED true [get_ports <list_of_ports>]
```



CAUTION! *For 7 series devices, Zynq-7000 UltraScale devices, UltraScale+ devices, and Zynq UltraScale+ MPSoCs, all I/O ports must have explicit values for the PACKAGE_PIN and IOSTANDARD constraints to generate a bitstream file. In the I/O Ports window, the word default is displayed in red to indicate that these values must be applied manually. You must apply extra care when assigning I/O standards, because these devices have Low and High voltage I/O banks.*

Setting I/O Port Direction

To set I/O port direction, use any of the following methods:

- For I/O planning projects only:
  - In the Direction column of the I/O Ports window, click a port and change the direction using the drop-down menu.
  - Click a port in the I/O Ports window, and change the direction in the I/O Port Properties window.
  - Select the I/O ports, buses, or interfaces to be configured in the I/O Ports window, right-click, and select **Set Direction**.
- For RTL projects only, define the port direction in the RTL source.

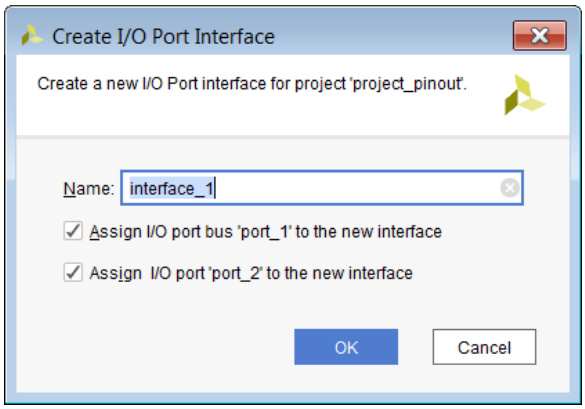
IMPORTANT! *You can only set the port direction property in an I/O planning project. If you attempt to modify this property outside of an I/O planning project, a critical warning is issued.*

Creating I/O Port Interfaces

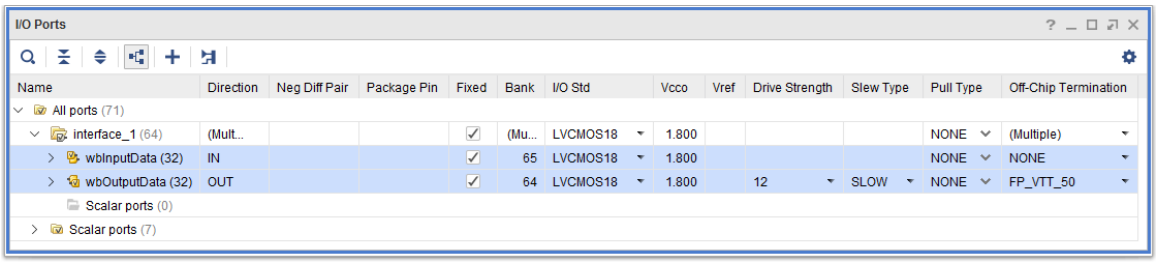
To group multiple ports or buses together, you can create an interface. This aids in pin assignment by treating all of the interface ports as one group. Assigning all of the pins simultaneously helps condense and isolate the interface for clock region or PCB routing. This also makes it easier to visualize and manage the signals associated with a particular logic interface.

To create an interface:

1. In the I/O Ports window, select the signals to group together.
2. Right-click, and select **Create I/O Port Interface**.
3. In the Create I/O Port Interface dialog box (see the following figure), enter a name for the interface, adjust assignment selection, and click **OK**.



The interfaces appear as expandable folders in the I/O Ports window as shown in the following figure.



TIP: *To delete interfaces, select an interface, right-click, and select* ***Delete****, or press the Delete key.*

Adding I/O Ports to an Interface

To add I/O ports to an interface, do either of the following in the I/O Ports window:

- Select the I/O ports, and drag them into the interface folder.
- Right-click a port or bus, and select **Assign to Interface**. In the Select I/O Port Interface dialog box, select the target interface.

Removing I/O Ports from an Interface

To remove I/O ports, in the I/O Ports window right-click a port, and select **Unassign from Interface**.

Tcl Command Examples for Working with I/O Port Interfaces

- Creating I/O port interfaces:

```
create_interface interface_1
set_property INTERFACE interface_1 [get_ports [list {test_1[3]}
{test_1[2]} \
{test_1[1]} {test_1[0]} {test_1_n[3]} {test_1_n[2]} {test_1_n[1]} \
{test_1_n[0]}]]
set_property INTERFACE interface_1 [get_ports [list port_2 port_2_N
port_1 \ port_4]]
```

- Removing I/O ports from an interface:

```
set_property INTERFACE "" [get_ports [list port_2 port_2_N]]
```

Automatically Inferring I/O Port Interfaces

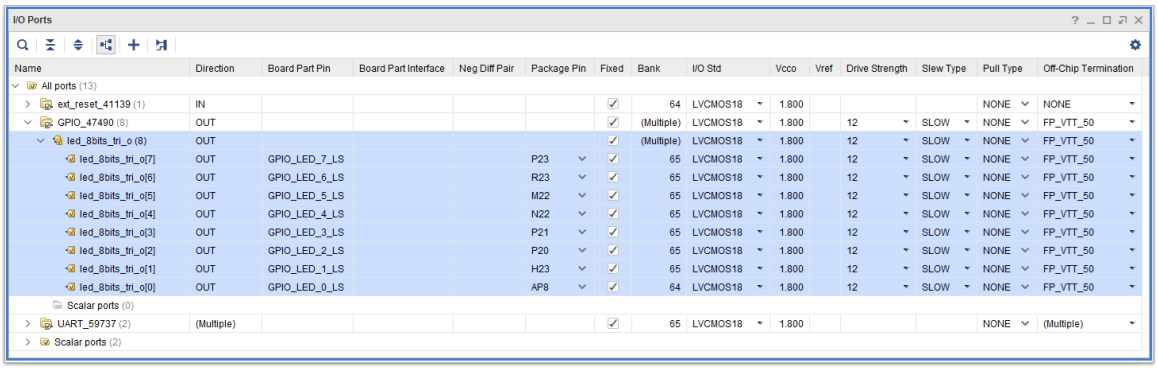
RECOMMENDED: *If your project targets a platform board rather than a part, Xilinx recommends that you use the Vivado Design Suite platform board flow to configure and apply board pinout constraints using the Board tab in the Customize IP dialog box or in the Board window in the Vivado IP integrator. For more information on the platform board flow, see this link in the Vivado Design Suite User Guide: System-Level Design Entry (UG895).*

You can view the interfaces that are connected from an IP to the top-level ports of your design. For these IP interfaces, the Vivado tools automatically infer a pin planning interface that groups the related top-level I/O ports. This provides a symbolic way of referring to the interface in the context of the top-level design. For example, in the following figure, the `led_8bits_tri_o` bus is a general purpose I/O (GPIO) interface that is grouped under the GPIO_9847 pin planning interface.

You can view the board part pins associated with the I/O ports from the Board Part Pin column in the I/O Ports window. In the following figure, the ports associated with pin-planning interface GPIO_9847 are constrained to board part pins `led_8bits_tri_o[7:0]`.

***Note*:** The number 9847 in GPIO_9847 is used for unique identification of the GPIO interface when there are multiple GPIO interfaces in the design. There is no specific meaning for the number.

Figure 16: **Automatic Inference of I/O Port Interfaces**



I/O Ports

Name	Direction	Board Part Pin	Board Part Interface	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength	Slew Type	Pull Type	Off-Chip Termination
All ports (13)														
ext_reset_41139 (1)	IN					✓	64	LVCMOS18	1.800				NONE	NONE
GPIO_47490 (8)	OUT					✓	(Multiple)	LVCMOS18	1.800		12	SLOW	NONE	FP_VTT_50
led_8bits_tri_o (8)	OUT					✓	(Multiple)	LVCMOS18	1.800		12	SLOW	NONE	FP_VTT_50
led_8bits_tri_o[7]	OUT	GPIO_LED_7_LS			P23	✓	65	LVCMOS18	1.800		12	SLOW	NONE	FP_VTT_50
led_8bits_tri_o[6]	OUT	GPIO_LED_6_LS			R23	✓	65	LVCMOS18	1.800		12	SLOW	NONE	FP_VTT_50
led_8bits_tri_o[5]	OUT	GPIO_LED_5_LS			M22	✓	65	LVCMOS18	1.800		12	SLOW	NONE	FP_VTT_50
led_8bits_tri_o[4]	OUT	GPIO_LED_4_LS			N22	✓	65	LVCMOS18	1.800		12	SLOW	NONE	FP_VTT_50
led_8bits_tri_o[3]	OUT	GPIO_LED_3_LS			P21	✓	65	LVCMOS18	1.800		12	SLOW	NONE	FP_VTT_50
led_8bits_tri_o[2]	OUT	GPIO_LED_2_LS			P20	✓	65	LVCMOS18	1.800		12	SLOW	NONE	FP_VTT_50
led_8bits_tri_o[1]	OUT	GPIO_LED_1_LS			H23	✓	65	LVCMOS18	1.800		12	SLOW	NONE	FP_VTT_50
led_8bits_tri_o[0]	OUT	GPIO_LED_0_LS			AP8	✓	64	LVCMOS18	1.800		12	SLOW	NONE	FP_VTT_50
Scalar ports (0)														
UART_59737 (2)	(Multiple)					✓	65	LVCMOS18	1.800				NONE	(Multiple)
Scalar ports (2)														

Tcl Command Examples for Working with the Platform Board Flow

- Getting board part interfaces:

```
get_board_part_interfaces
```

- Getting ports associated with the board interface:

```
get_board_interface_ports -of [get_board_part_interfaces]
```

- Getting properties of the port:

```
get_property NAME [lindex [get_board_interface_ports] 5]
get_property DIRECTION [lindex [get_board_interface_ports] 5]
```

- Getting properties from the board part pin:

```
get_property IOSTANDARD [get_board_part_pins]
get_property LOC [get_board_part_pins]
```

Note: For information on the `get_board_part_interfaces` Tcl command and other Tcl commands, see the *Vivado Design Suite Tcl Command Reference Guide* (UG835).

Prohibiting I/O Pins and I/O Banks

The I/O Planning view layout provides an interface to selectively prohibit port placement onto individual I/O pins, groups of I/O pins, or I/O banks. You can select and prohibit pins in the Device, Package, and Package Pins windows.

To prohibit I/O pins or I/O banks:

1. Select the I/O pins or I/O banks in the Device, Package, or Package Pins window.
2. Right-click, and select **Set Prohibit**.

Prohibited pins are indicated by:

- Slashed circle in the Device window and Package window (see the following figure)

- Check mark in the Prohibit column of the Package Pins window

TIP: *You can clear prohibits from the Prohibit column of the Package Pins view. Select an individual prohibit or use Ctrl+A to select all pins, right-click, and select* ***Clear Prohibit****. Alternatively, you can use a Tcl command to clear the prohibit, for example:* `set_property prohibit 0 [get_sites U17]`.

Tcl Command Example for Prohibiting I/O Pins

```
set_property PROHIBIT 1 [get_sites U17]
```

Placing I/O Ports

The I/O Planning view layout provides several ways to assign I/O ports to package pins. You can select individual I/O ports, groups of I/O ports, or interfaces in the I/O Ports window, and assign them to package pins in the Package window or to I/O pads in the Device window.

In the Package window, you can:

- Drag and drop ports to package pins.
- View port placement and constraints.
- Move the cursor over the pins to show the I/O pin coordinates on the top and left sides of the window.
- Hold the cursor over a pin to show a tooltip that displays the pin information.
- Display differential pairs by selecting the Package window settings and opening the General tab.

Note: Additional I/O pin and bank information displays in the status bar located at the bottom of the Vivado IDE.

Placing I/O Ports Sequentially

To place I/O ports sequentially:

1. In the I/O Ports window, select an individual I/O port, a group of I/O ports, or interfaces.
2. Use one of the following commands:
   - In the I/O Ports window, right-click, and select **Place I/O Ports Sequentially**.
   - In either the Package window or the Device window, right-click, and select **Place I/O Ports Sequentially**.

   The first I/O port in the group is attached to the cursor when you move it over a package pin or I/O pad. A tooltip displays the I/O port and package pin names.
3. To assign an I/O port, click a pin or a pad.

   If you select more I/O ports, the command is continued. The cursor drags the subsequent I/O ports until all of the I/O ports are placed, or you press **Esc**.

> **TIP:** *The Vivado IDE assigns ports in the order that they appear in the I/O Ports window. You can adjust the assignment order by applying sorting techniques in the I/O Ports window prior to assignment.*

Placing I/O Ports into I/O Banks

To place I/O ports into I/O banks:

1. In the I/O Ports window, select an individual I/O port, a group of I/O ports, or interfaces.
2. Use one of the following commands:
   - In the I/O Ports window, right-click, and select **Place I/O Ports in an I/O Bank**.
   - In either the Package window or the Device window, right-click, and select **Place I/O Ports in an I/O Bank**.

     The group of I/O ports is attached to the cursor when it is dragged over a package pin or I/O pad. A tooltip shows the number of pins that can be placed in the selected I/O bank.
3. Click a pin or pad to assign the selected I/O ports.

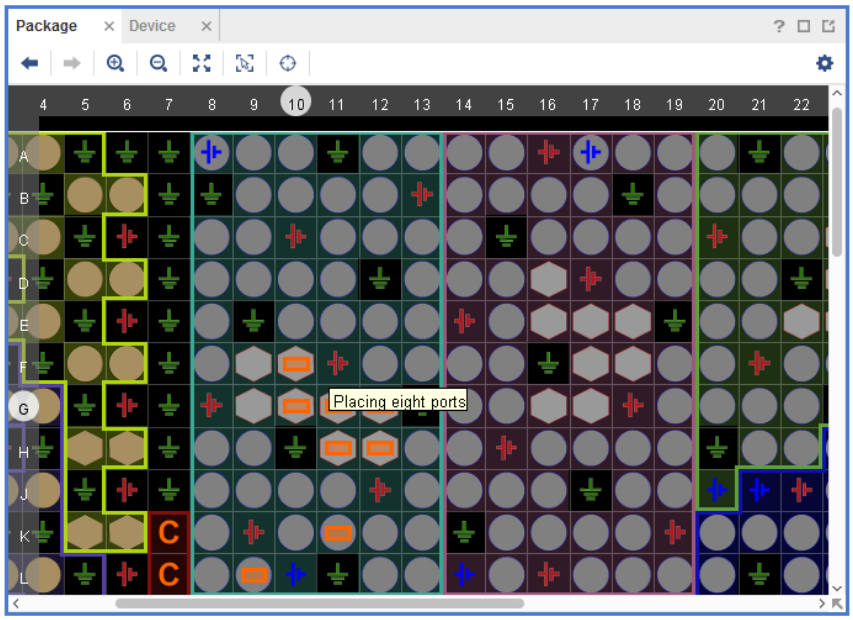
   If more I/O ports are selected than fit in the I/O bank, the Vivado IDE places as many as possible in the selected I/O bank, then lets you select another I/O bank into which to place the remaining ports. The cursor drags the remaining I/O ports to the next selected I/O bank until all of the I/O ports are placed, or you press **Esc**.

> **TIP:** *The Vivado IDE assigns ports in the order that they appear in the I/O Ports window. You can adjust the assignment order by applying sorting techniques in the I/O Ports window prior to assignment.*

Port assignment to device resources is also driven from the initial selection from the I/O bank. Selecting a pin at one end of an I/O bank results in a continuous bus assignment across the I/O bank.

The Vivado IDE also keeps track of PCB routing concerns for buses. Pin ordering during assignment attempts to keep the bus bits vectored within the assignment area. You can customize assignment patterns to address other bus routing concerns.

The following figure shows I/O ports placed in an I/O bank.



Tcl Command Examples for Placing Ports into I/O Banks

- Placing ports into I/O banks:

```
place_ports -iobank [get_iobanks {12 13 14 15}] [all_inputs]
```

- Placing ports into all I/O banks:

```
place_ports -iobank [lrange [get_iobanks] 1 end] <port list>
```

Note: The `place_ports` command is not supported for bank 0.

Placing I/O Ports in a Defined Area

To place I/O Ports into a defined area:

1. In the I/O Ports window, select individual I/O ports, groups of I/O ports, or interfaces.
2. Use one of the following commands:
   - In the I/O Ports window, right-click, and select **Place I/O Ports in Area**.

- In either the Package window or the Device window, right-click, and select **Place I/O Ports in Area**.

  The cursor turns into a cross symbol, which indicates that you can define a rectangle for port placement.

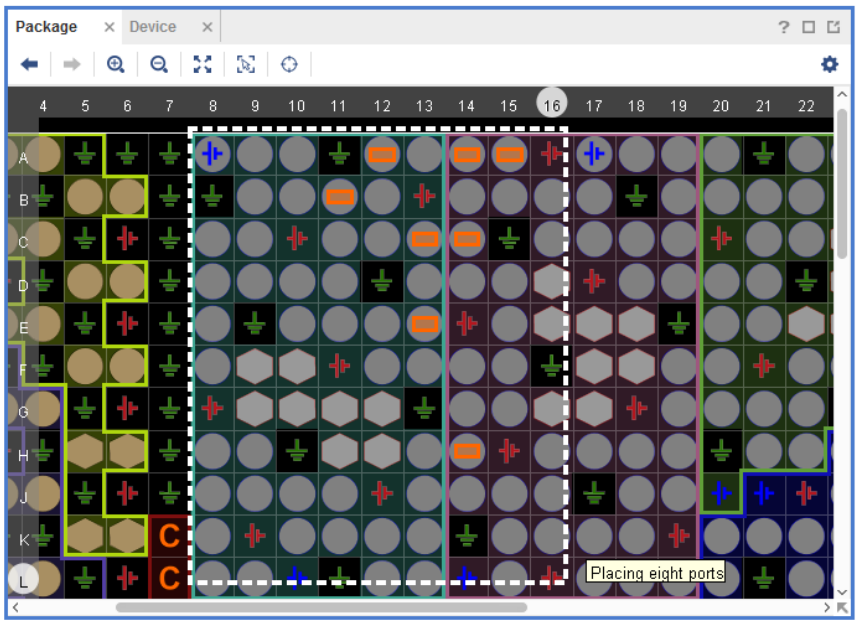
3. In either the Package window or the Device window, draw a rectangle to define the assignment area.

   If you select more I/O Ports than fit in the defined area, the command is continued. The cursor continues to display as a cross to draw another area to place the remaining I/O ports until all of the I/O ports are placed, or you press **Esc**.

   > **TIP:** *The Vivado IDE assigns ports in the order that they appear in the I/O Ports window. You can adjust the assignment order by applying sorting techniques in the I/O Ports window prior to assignment.*

   The direction in which you draw the rectangle dictates the I/O ports assignment order. I/O ports are assigned from the inside pin of the first rectangle coordinate selected. Creative definition of the area rectangles can provide useful pinout configurations from a PCB routing perspective.

   The following figure shows I/O ports placed in an area.



Swapping Previously Placed I/O Ports

To swap the location of two placed I/O ports that are already assigned:

1. Select two I/O ports from any of the available windows.
2. Right-click, and select **Swap Locations**.

IMPORTANT! *If you are working in an implemented design and you swap two ports that are not yet fixed, swapping the ports fixes the ports and writes constraints to the XDC file.*

Moving Previously Placed I/O Ports

To move a port or group of ports that are already assigned, select the port or group of ports, and drag them from one location to another. When you move a group of ports from one I/O bank to another, the Vivado IDE automatically finds suitable locations for the selected ports.

***Note*:** This is similar to using the Place I/O Ports in an I/O Bank command.

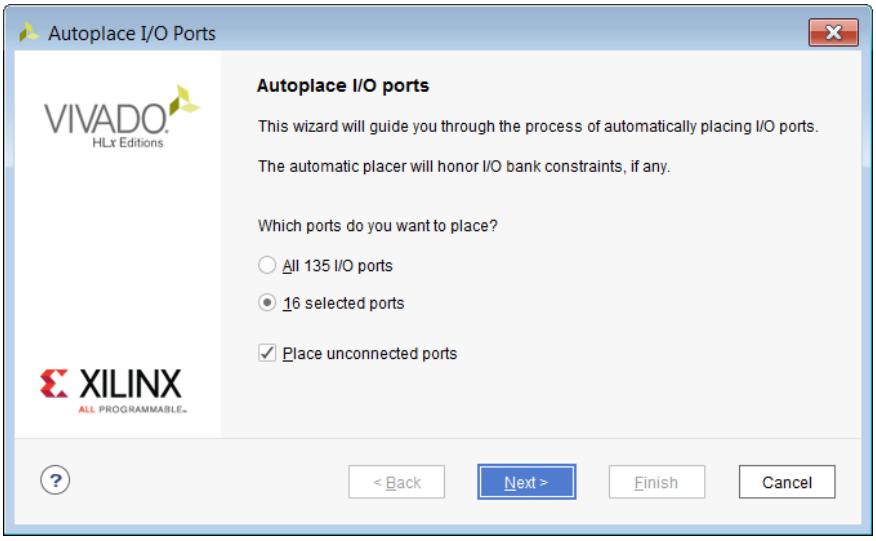
Automatically Placing I/O Ports

You can automatically assign I/O ports to package pins in an open synthesized design. The Vivado IDE obeys I/O standard and differential pair rules and places global clock pins appropriately.

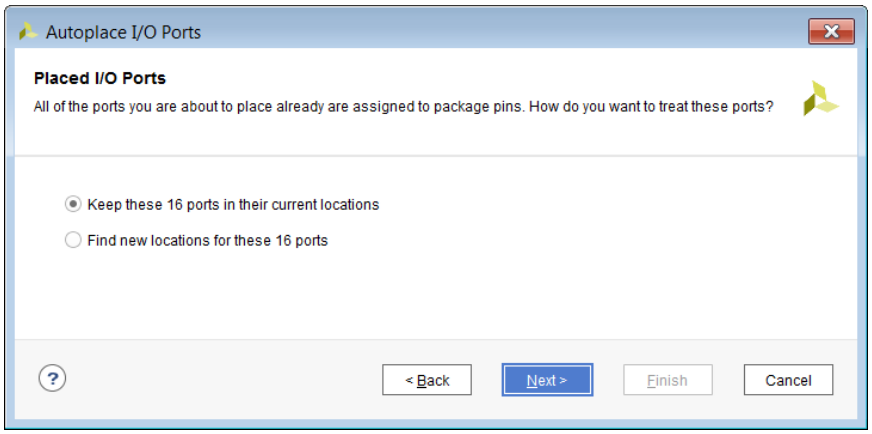
To automatically assign I/O ports:

1. In the I/O Ports window, select the I/O ports to place.
2. Select **Tools → I/O Planning → Auto-place I/O Ports**.

   ***Note*:** Alternatively, you can right-click, and select **Auto-place I/O Ports** in the I/O Ports window.
3. In the Autoplace I/O Ports wizard (see the following figure), select the group of I/O ports to place, and click **Next**.



4. If you selected I/O ports that are already assigned to package pins, select an option in the Placed I/O Ports page, and click **Next**.



5. In the Summary page, click **Finish**.

Placing Gigabit Transceiver I/O Ports

To better manage GTs, the I/O planning windows group the two related I/O diff pairs and the GT logic object automatically during selection, placement, and moving. The GT objects are selected as one object and move together, which prohibits illegal assignment of the GT resources.

If the interactive DRCs are enabled, the noise sensitive I/O pins surrounding the GTXs are prohibited automatically during port placement. For more information, see Enabling or Disabling Interactive DRCs.

For information about transceiver placement rules, see the following, depending on your device.

- *7 Series FPGAs GTX/GTH Transceivers User Guide* (UG476)
- *UltraScale Architecture GTH Transceivers User Guide* (UG576)

TIP: *When placing gigabit ports for 7 series, UltraScale, and UltraScale+, place them sequentially by right-clicking the ports that need to be placed, selecting Place I/O Sequentially, and then moving the mouse over the ports in the Package window.*

RECOMMENDED: *Xilinx recommends that you begin pin planning for gigabit transceivers at IP customization. If you are using IP integrator, begin pin planning at that time.*

Related Information

Enabling or Disabling Interactive DRCs

Removing I/O Placement Constraints

To remove placement constraints, select the placed logic, right-click, and select **Unplace**.

Migrating an I/O Planning Project to an RTL Project

After the I/O ports are defined and placed onto the package pins, you can migrate the I/O planning project to an RTL project. The port definitions are used to create a top module for the RTL design in either Verilog or VHDL, as specified. Differential pair buffers are added to the top module, and bus definitions are also included in the RTL. The project properties are changed to reflect the RTL project type.

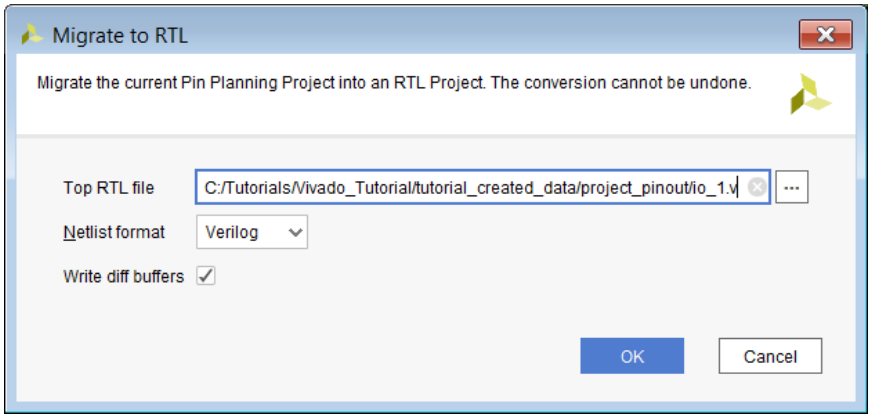
IMPORTANT! *After migration, the RTL project cannot be converted back into an I/O planning project.*

To convert the project:

1. Select **File → Migrate to RTL**.

   Note: Alternatively, you can select **Migrate to RTL** from the Flow Navigator.

2. In the Migrate to RTL dialog box (see the following figure), set the following options, and click **OK**.
   - **Top RTL file:** Specify the Verilog (`.v` extension) or VHDL (`.vhd` extension) file to create for the top module of the design. The HDL file will include the module definition with port definitions, direction, and width for bus pins.
   - **Netlist format:** Specify Verilog or VHDL format for the top module.
   - **Write diff buffers:** Write the diff pair buffers as part of the top module definition. This preserves any differential pairs defined in the I/O planning project.



After the I/O planning project is converted to an RTL project, you can begin adding sources to the project and working on your design. For more information, see this link in the *Vivado Design Suite User Guide: System-Level Design Entry* (UG895).

Chapter 4

I/O Planning for UltraScale Architecture Memory IP

UltraScale™ architecture Memory IP defines a memory controller using a pre-engineered controller and physical layer (PHY) for interfacing FPGA user designs and AMBA® specification advanced extensible interface (AXI4) slave interfaces to supported external memory devices. High-speed memory interfaces must adhere to:

- Specific pinout requirements driven by clocking and skew needs
- Specific rules for byte lane usage within the I/O banks for memory
- Physical pin assignment requirements

Memory also has hard memory controller only I/O pins (DDRMC). These pins cannot be used for anything other than the hard memory controller. So if the design does not use these pins, those pins cannot be used for GPIO.

For performance purposes, the final configuration of the Memory IP is dependent on the I/O assignments. Therefore, you cannot complete final implementation of the IP until after the IP's I/Os are assigned. For this reason, you must handle the I/O assignment and implementation of this IP differently from most other IP. This chapter describes the process for I/O planning and implementation of UltraScale architecture Memory IP.

RECOMMENDED: *Due to the restrictions related to port grouping and I/O bank assignments for memory controllers, Xilinx recommends that you complete I/O planning for memory controllers before general I/O assignment in a post-synthesis project.*

VIDEO: *For more information, see the Vivado Design Suite QuickTake Video: Designing with UltraScale Device Memory IP.*

IMPORTANT! *This chapter covers UltraScale architecture Memory IP only. For information on Memory IP for 7 series devices, see the Zynq-7000 SoC and 7 series Devices Memory Interface Solutions (UG586).*

UltraScale Architecture Memory IP I/O Planning Design Flow Changes

The Vivado® Design Suite has the following differences between the I/O assignment and implementation process for UltraScale architecture Memory IP:

- Consolidated Memory IP I/O planning with the rest of the design in the main Vivado IDE I/O Planning view layout, which enables pin planning with the design RTL or after synthesizing the design.
- PHY implementation of the IP now performed after synthesis as a part of the `opt_design` command, which enables netlist based I/O planning.
- Physical block (Pblock) that contains the IP now automatically generated as a part of the `opt_design` command and is transient and invisible to users.

Consolidated I/O Planning

In previous Vivado Design Suite releases, you made all I/O assignments as a part of customizing the IP. The tools stored the resulting constraints with the IP in a read-only XDC file. This required you to re-customize the IP to modify the port assignments. In addition, these constraints were not necessarily visible during I/O assignment and validation for the rest of the design. Starting in the 2015.x releases, you can perform memory I/O assignment in the main Vivado IDE I/O Planning view layout along with the rest of the design ports. It is no longer a part of Memory IP configuration. This tool is called the Advanced I/O Planner and now includes XPIO SelectIO interfaces. It places all the I/O at once to optimize I/O placement and packing.

The new Memory I/O methodology enables you to:

- Make changes to memory I/O ports without regenerating Memory IP.
- Target Memory IP to different devices with different pinouts without regenerating Memory IP.
- Perform I/O planning with multiple memory controllers concurrently in one environment.
- Define and store memory port assignments in the top-level XDC constraints file for the design rather than in a read-only file within the IP.
- Directly edit or replace the XDC constraint file or files to modify memory I/O port assignments.

PHY Implementation

UltraScale architecture Memory IP defines a memory controller using a pre-engineered controller and PHY for interfacing user designs and AMBA specification AXI4 slave interfaces to DDR3, DDR4, QDRII+, QRDRIV, and RLDRAM3 SRAM devices.

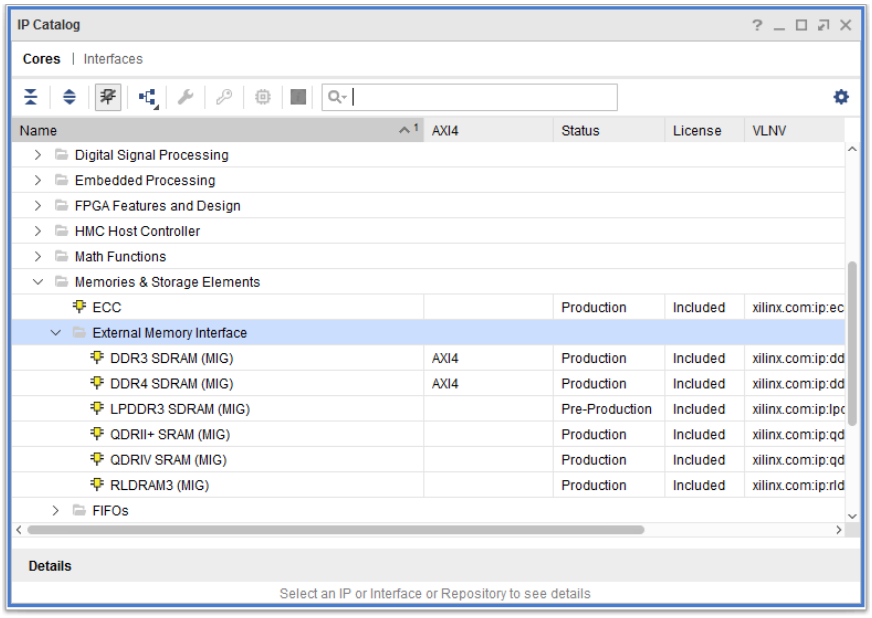
The Memory IP is structured so that only the physical layer (PHY) interconnect in the Xilinx® device needs to be updated when pinouts change. Because the PHY implementation depends on the I/O assignments, it must occur *after* the I/Os are placed and validated. To enable memory I/O planning after synthesis, the implementation of the PHY now happens as a part of implementation during the `opt_design` command.

Configuring Memory IP

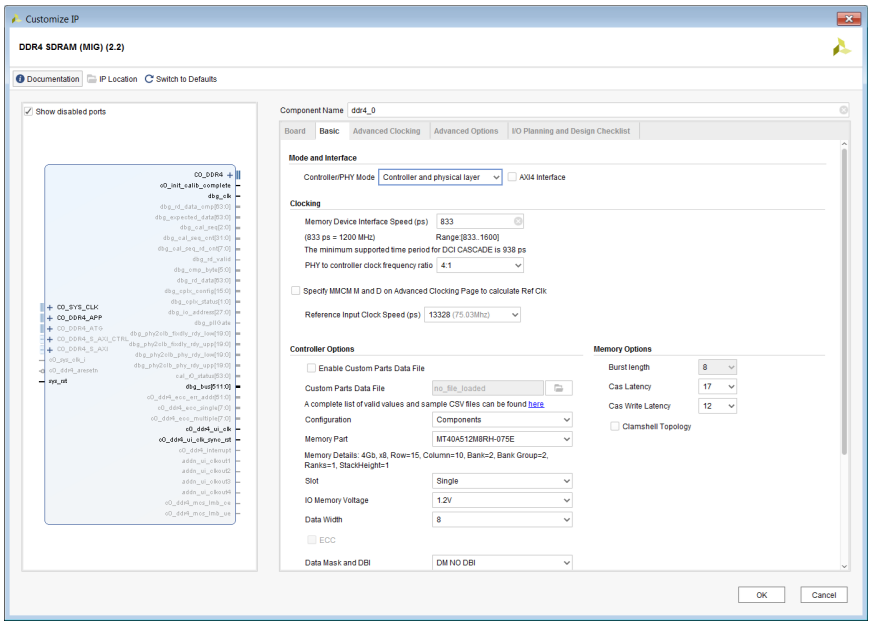
The UltraScale architecture Memory IP enables configuration of DDR3 and DDR4 SDRAM, QDRIIPLUS SRAM, and RLDRAM3 type interfaces. As of 2015.3, the Memory IP has been split into different IPs based on memory interface standards and tool flows. The Customize IP dialog box contains basic and advanced configuration options that include debug. Now that the memory I/O assignment process is consolidated with the rest of the design, the IP configuration process is consistent with other Xilinx IP. For more information on the IP configuration and management process, see this link in the *Vivado Design Suite User Guide: Designing with IP* (UG896).

To configure the Memory IP in the Vivado tools:

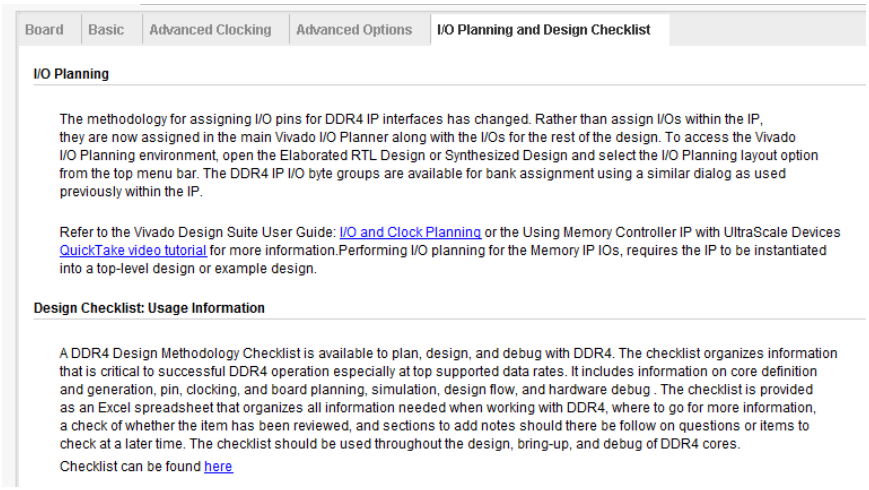
1. Open the Vivado IP catalog, and expand the **Memories & Storage Elements → External Memory Interface** category.



2. Double-click the desired interface to open the Customize IP dialog box.



***Note*:** Although the I/O Planning tab still exists in the Customize IP dialog box, it only explains the new consolidated Memory IP I/O planning.



For information on configuring Memory IP, see the *UltraScale Architecture-Based FPGAs Memory IP LogiCORE IP Product Guide* (PG150).

***Note*:** The Vivado Design Suite supports multiple UltraScale architecture memory controllers in the same design. Each must be defined individually.

Generating IP Output Products

After the Memory IP is configured, you must generate output products for use in implementation and third-party tools. When you generate output products, the Vivado tools create an IP-level XDC constraints file in the IP directory. The XDC file contains I/O physical constraints, such as IOSTANDARD, OUTPUT_IMPEDANCE, DRIVE, and SLEW. The Vivado tools derive these constraints from the configuration settings in the IP to aid in I/O planning. The constraints appear in the I/O Planning view layout.

For more information on generating output products, see this link in the *Vivado Design Suite User Guide: Designing with IP* (UG896).

UltraScale Architecture Memory IP I/O Planning in the Vivado IDE

If your design contains UltraScale architecture Memory IP, the Vivado IDE includes the following special features:

- Groups I/O ports for each Memory IP into port interfaces in the I/O Ports window, which enables group selection and modification
- Prevents all Memory IP-related ports from using the interactive port placement features, such as drag and drop, swap ports, or manual moving of ports in the graphical views
- Provides the Memory Bank/Byte Planner, which allows automatic or manual assignment of memory I/O pin groups to I/O banks and byte lanes

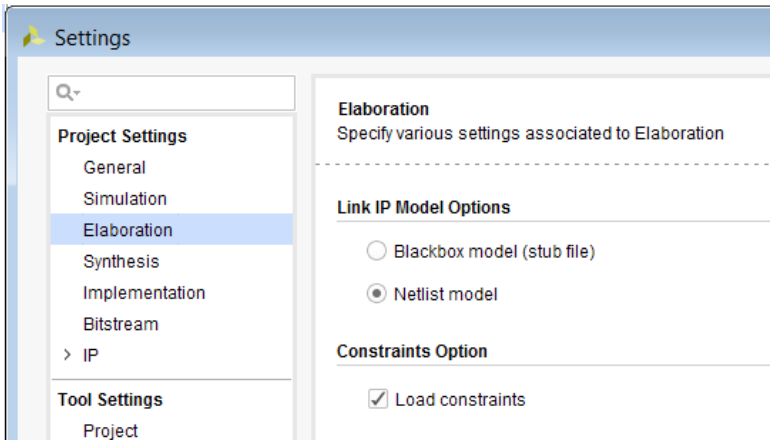
You can perform interactive I/O planning by opening either the elaborated RTL design or the synthesized design in the Vivado IDE. For both elaborated and synthesized designs, you can use the same basic process and commands. However, the Vivado tools perform more detailed DRCs in the synthesized design.

UltraScale Architecture Memory IP I/O Planning in the Elaborated Design

When using the elaborated design for memory I/O planning, you must set the proper Elaboration options before opening the design as follows:

1. In the Vivado IDE, select **Flow → Elaboration Settings**.
2. In the Project Settings dialog box, ensure that the **Netlist model** and **Load constraints** options are selected.

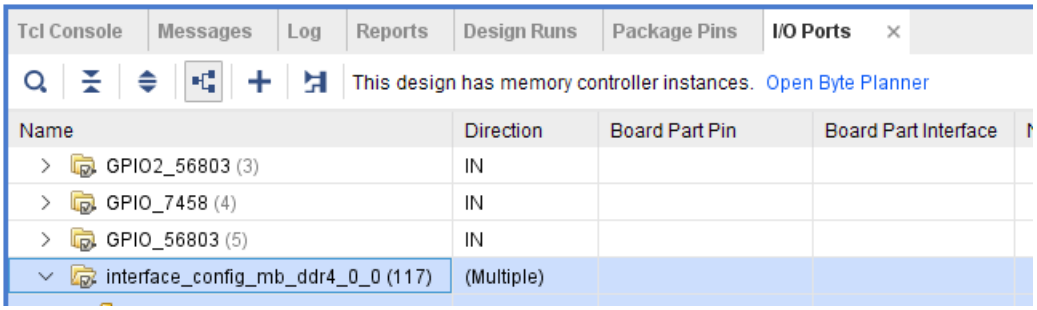
Loading the netlist model allows the elaborated design to read the synthesized Memory IP with the selected I/O properties, such as IOSTANDARD and OUTPUT_IMPEDANCE. If you use the black box model, you *cannot* do Memory IP I/O planning in the elaborated design.



Using the Memory Bank/Byte Planner

The I/O Planning view layout includes the I/O Ports and Package Pins windows. If UltraScale architecture Memory IP exists in the design, the banners of both windows contain a message and a button to launch the Memory Bank/Byte Planner.

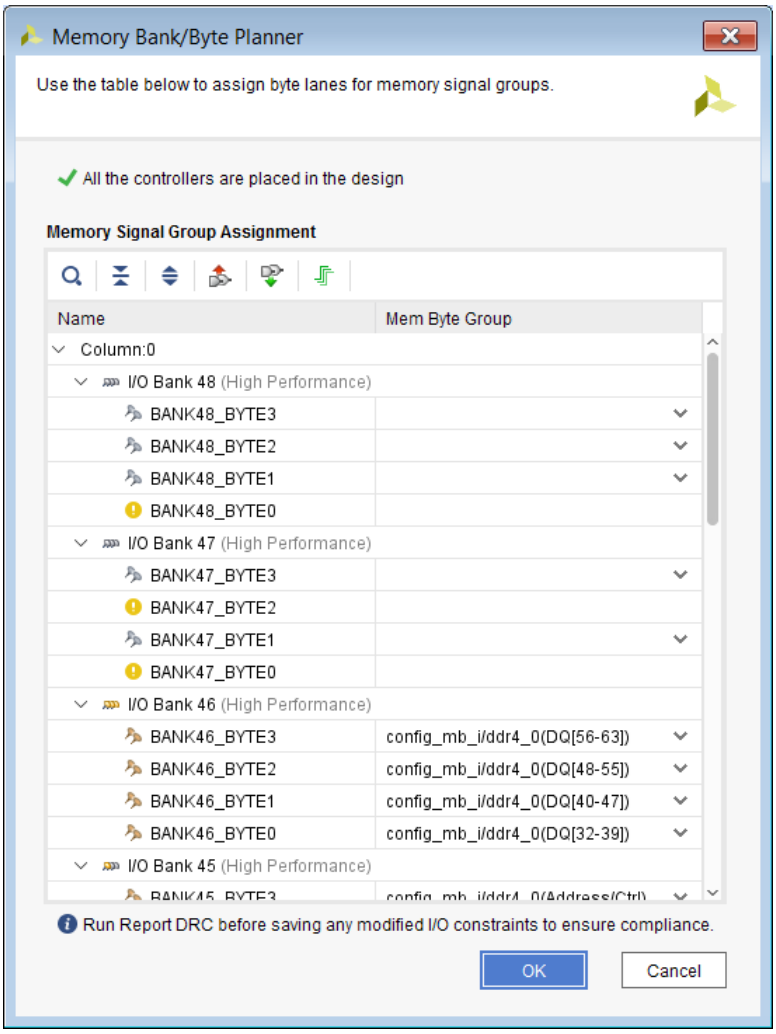
Figure 17: **Invoking the Memory Bank/Byte Planner**



***Note*:** Alternatively, you can open the Memory Bank/Byte Planner by selecting **Tools → I/O Planning → Memory Byte Planner**.

You can use the Memory Bank/Byte Planner to either automatically or manually assign memory interface signal groups to specific byte groups within the I/O banks.

Figure 18: **Memory Bank/Byte Planner**

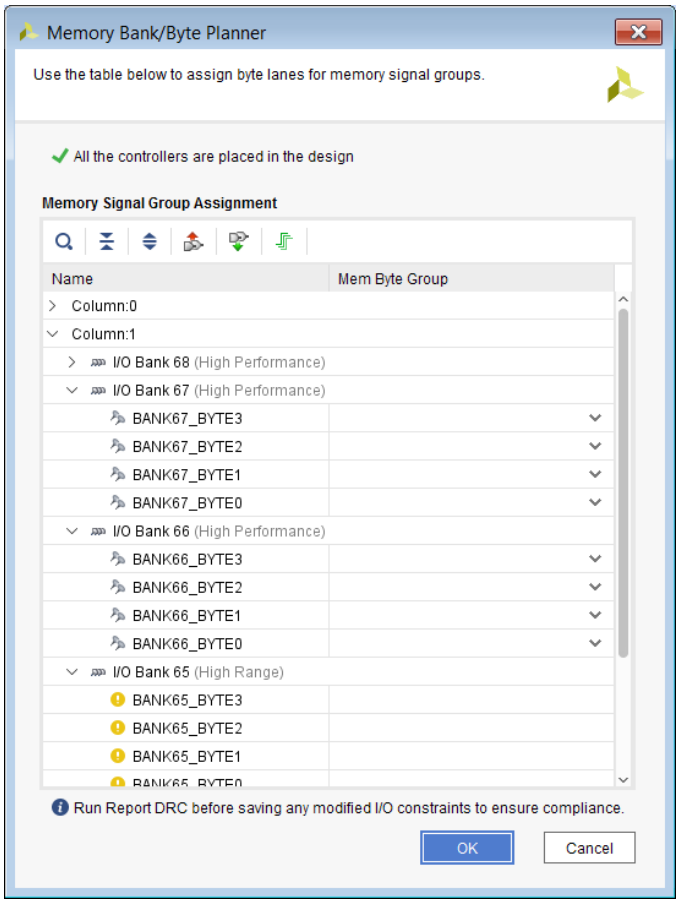


Note: If you use example designs generated directly from the Memory IP, the XDC file in the example design provides default I/O assignments that appear in the Memory Bank/Byte Planner.

The Memory Bank/Byte Planner includes the following features:

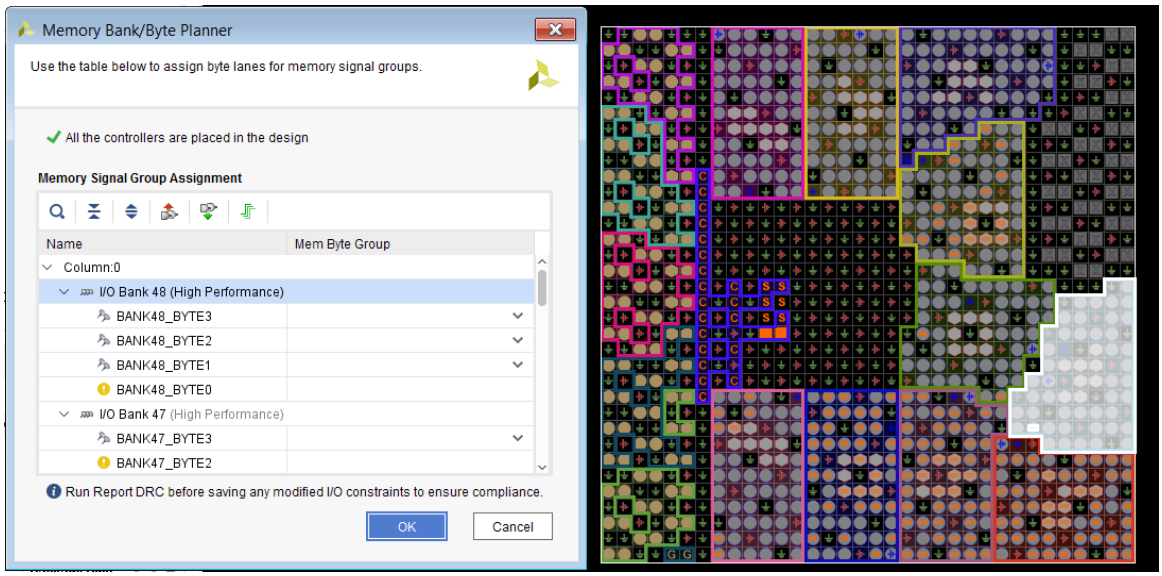
- **Collapsible device resource tree:** Device resources, such as super logic regions (SLRs), I/O columns and banks, and byte groups, appear in a collapsible and expandable tree that varies depending on the selected device. You can collapse the tree to target a specific area of the device, as shown in the following figure. The tree shows the resources in the order they appear on the device, because memory interfaces must be assigned to adjacent I/O banks.

Figure 19: **Collapsing the Device Resource Tree**



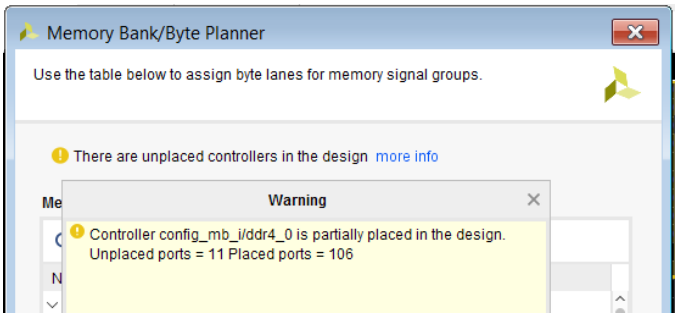
- **Cross-selection with other views:** When you select I/O banks and byte groups, the groups are also highlighted in the Package and Device windows to aid in identifying the resources, as shown in the following figure.

Figure 20: **Cross-Selecting Banks and Byte Groups**



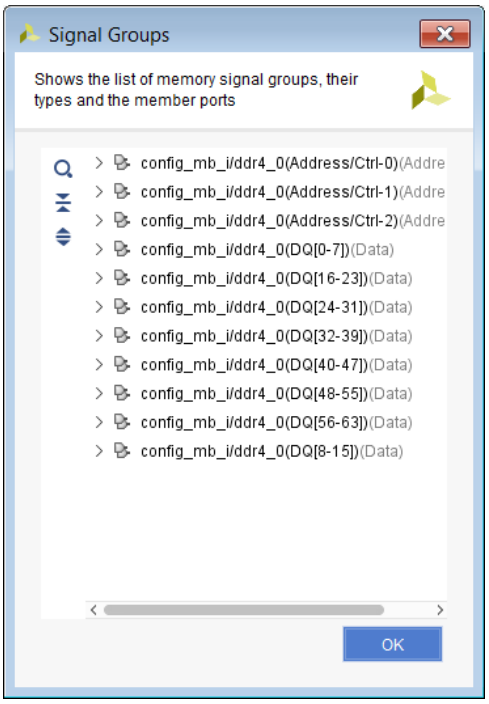
- **DRC information:** At the top of the Memory Bank/Byte Planner (following figure), a DRC status message provides information about DRC violations with a link to more information. When you select I/O banks and byte groups, the groups are also highlighted in the Package and Device windows to aid in identifying the resources, as shown in the following figure.

Figure 21: **Showing DRC Violations**



- **Signal group information:** Click the **Show Signal Group** button to show the list of signal groups for each Memory IP in the Signal Groups dialog box.

Figure 22: **Signal Groups Dialog Box**

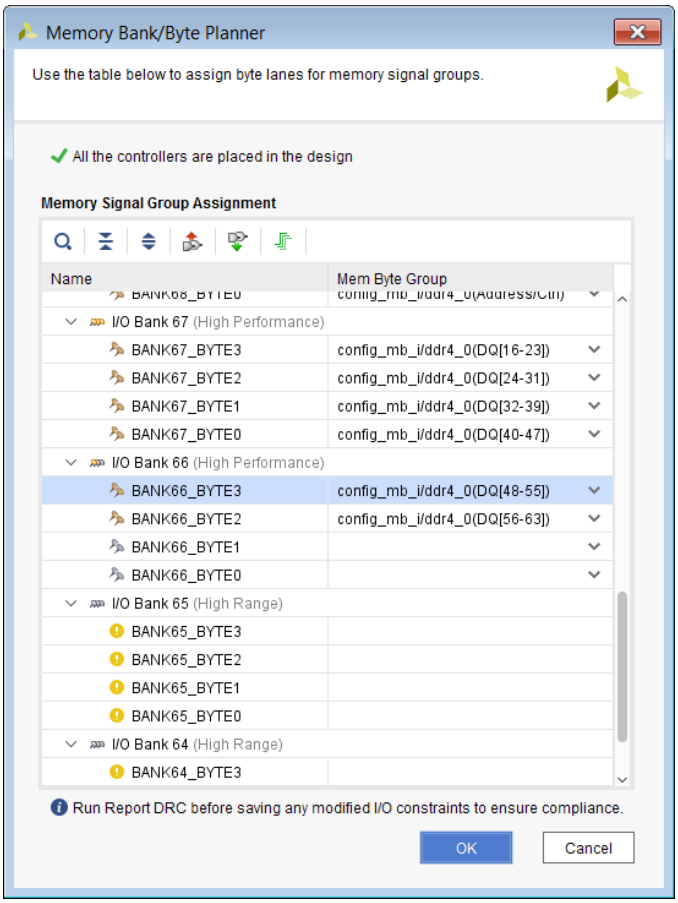


Manually Assigning Signal Groups

To manually assign signal groups to byte lanes:

1. In the Mem Byte Group column in the following figure, click the drop-down list next to a bank.
2. Select a signal group to assign.

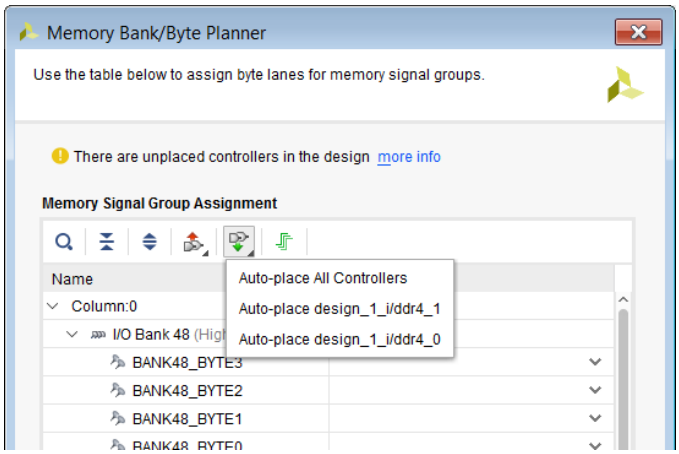
After each assignment, the Vivado tools run active DRCs. DRC violations appear in red, and you can click the **more info** link for details. The Vivado IDE shows signal groups for each Memory IP in the design, so you can plan I/O assignments for multiple memory controllers at the same time.



Automatically Assigning Signal Groups

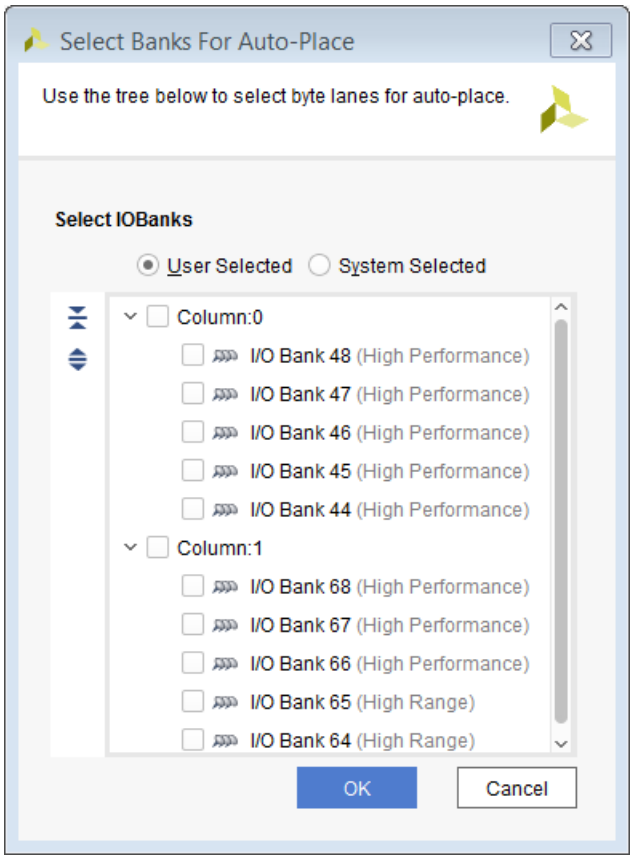
You can automatically place the signal groups for memory controllers individually or all at one time. You can target an I/O bank or a group of I/O banks for each memory controller. To automatically assign signal groups:

1. Click the **Auto Assign Controllers** button .
2. Select **Auto-place All Controllers** or **Auto-place <controller name>**.



3. In the Select Banks for Auto-Place dialog box, select one of the following options:
   - **System Selected:** Allows the Vivado tools to find an optimal placement for the memory controllers.
   - **User Selected:** Allows you to select the banks to target for the memory controllers.

   Note: A warning opens if you did not select enough I/O banks to accommodate the memory controllers.

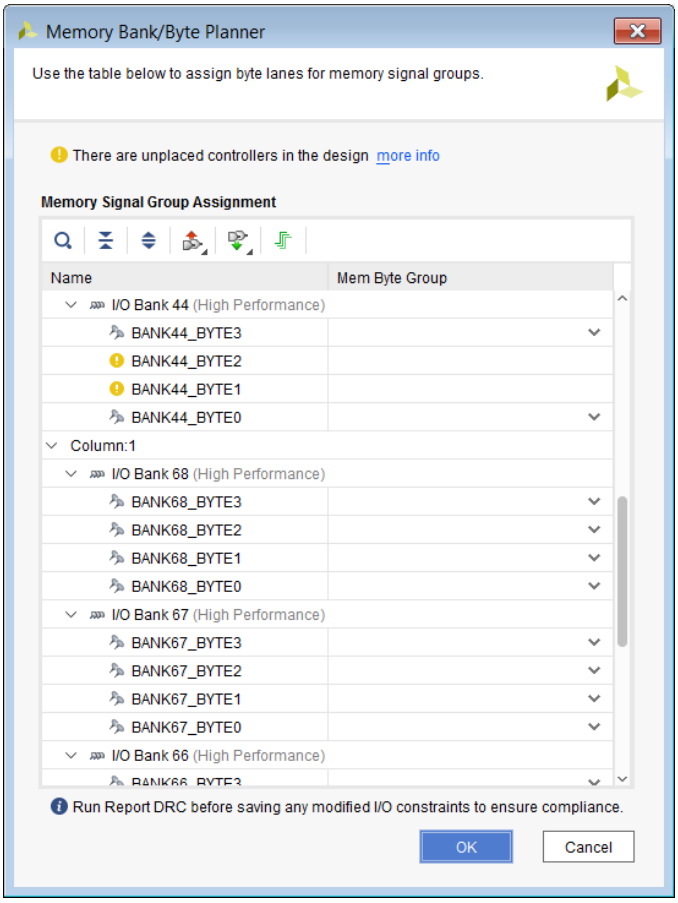


After auto-placing, a confirmation dialog box lists the number of placed ports.

Understanding Warnings for Assigned Non-Memory Ports

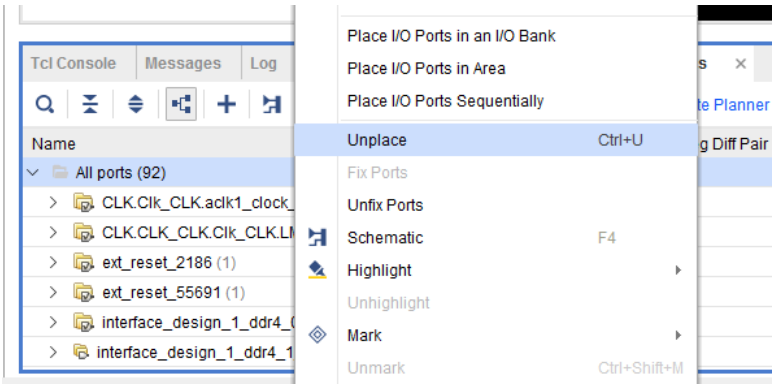
The Memory Bank/Byte Planner does not allow signal groups to be assigned to I/O banks that are already assigned to non-memory I/O ports. A yellow warning icon appears with a tooltip indicating that non-memory ports are assigned to the byte group, as shown in the following figure.

Figure 23: **Warnings for Assigned Non-Memory Ports**



You can address this issue by unplacing the ports. In the I/O Ports window, right-click the ports, and select **Unplace**, as shown in the following figure.

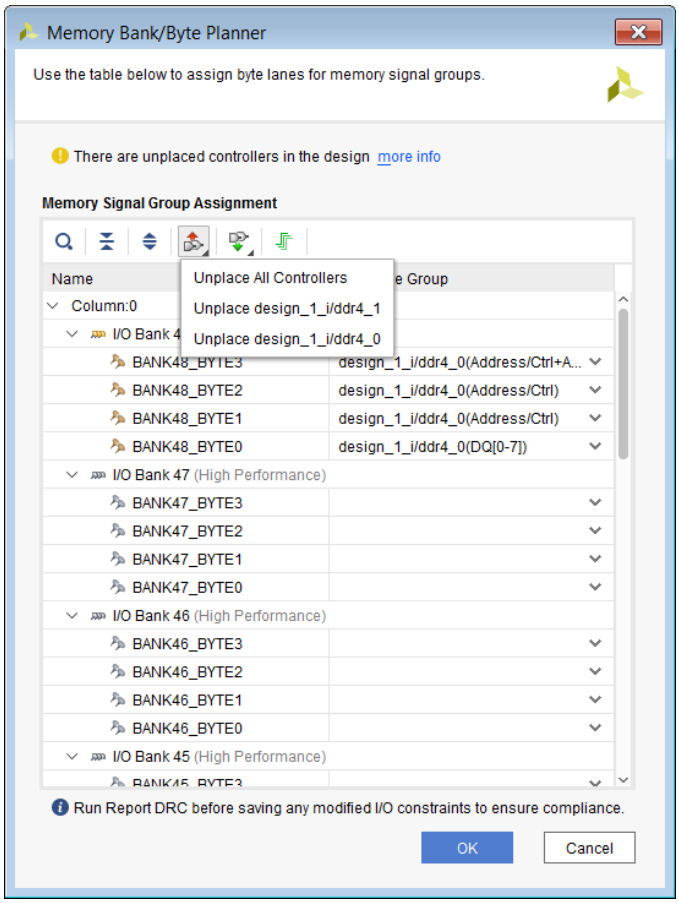
Figure 24: **Unplacing Non-Memory Ports**



Unplacing Signal Groups and Controllers

You can unplace the signal groups for memory controllers individually or all at one time. To unplace signal groups:

1. Click the **Unplace Ports for Memory Controllers** button .
2. Select **Unplace All Controllers** or **Unplace <controller name>**.



Modifying Memory I/O Ports

You can manually modify memory I/O ports, such as swap pins, using the I/O Ports window or the Package Pins window. Both windows are expandable data tables, which you can manipulate to more easily view information. For example, you can expand or collapse the table, filter lists, or sort and move columns. For more information, see this link in the *Vivado Design Suite User Guide: Using the Vivado IDE* (UG893).

IMPORTANT! *DRCs are not performed as you make changes, which means it is possible to make invalid pin assignments for the memory controller. After you modify the I/O ports or package pin assignments, it is important to run DRCs as described in Running DRCs.*

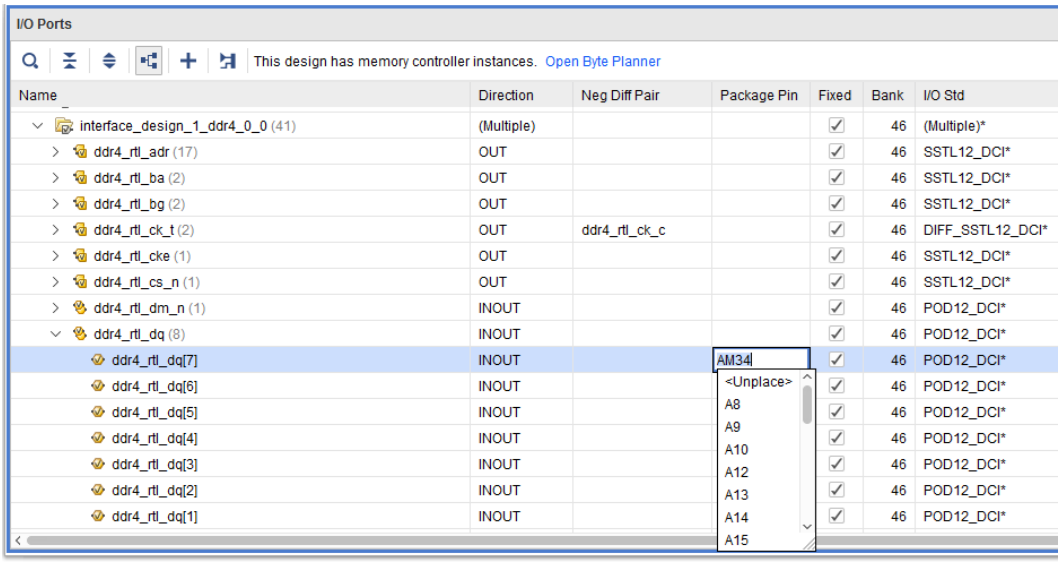
Related Information

Running DRCs

Modifying Ports in the I/O Ports Window

To modify port assignments in the Package Pins window, do one of the following in the Ports column:

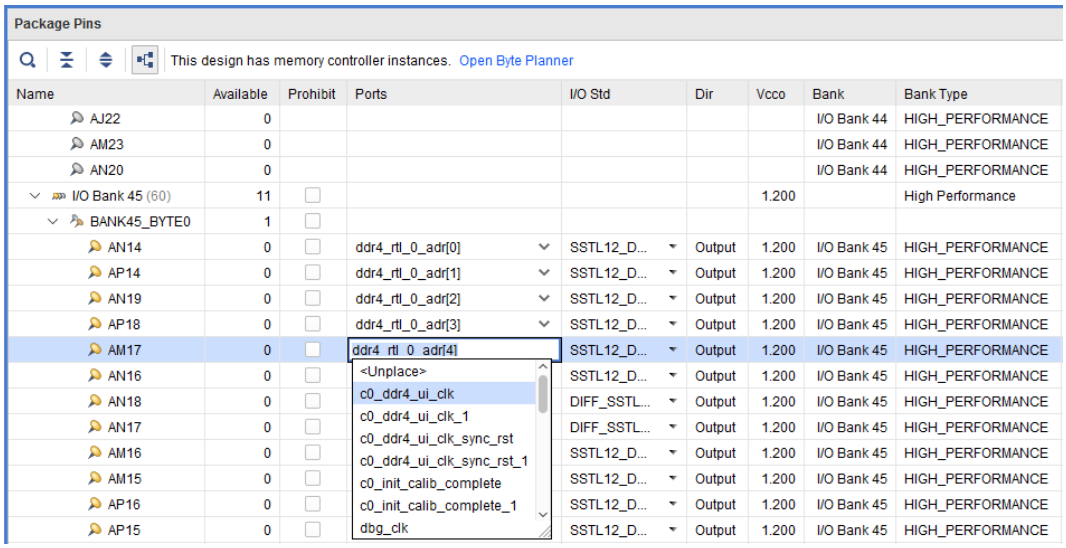
- Select a signal from the drop-down list.
- Type a signal name.



Modifying Ports in the Package Pins Window

To modify port assignments in the Package Pins window, do one of the following in the Ports column:

- Select a signal from the drop-down list.
- Type a signal name.



Running Memory DRCs

After assigning or modifying pin assignments, you must run the default DRC rule deck on the elaborated or synthesized design using the Report DRC command as described in Running DRCs.

Note: More design rules are available in the synthesized design, because the netlist is complete.

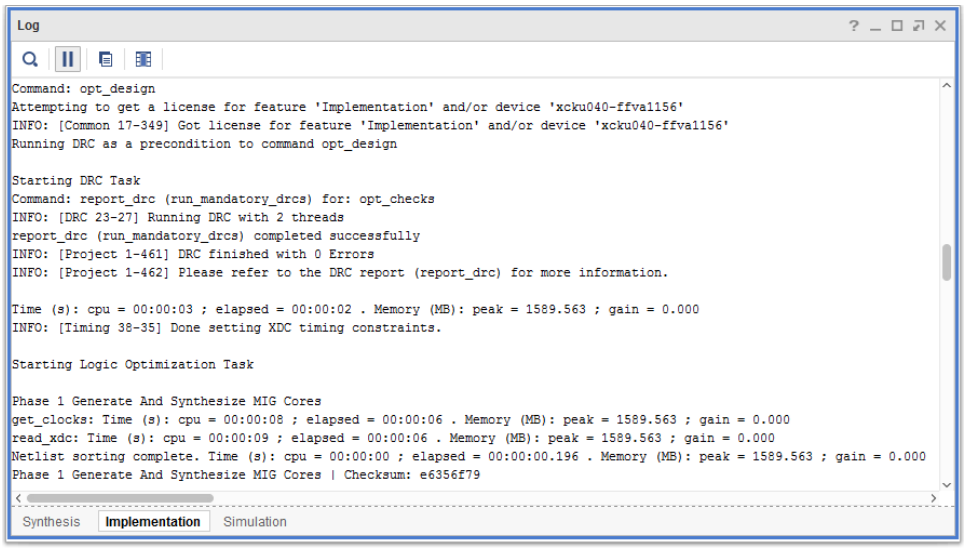
Related Information

Running DRCs

Implementing the PHY

For each memory controller, the Vivado tools synthesize and stitch the physical layer (PHY) into the netlist during implementation when Phase 1 of the `opt_design` command is run, as shown in the following figure.

Figure 25: **Implementing PHY During opt_design**



Tcl Command Example for Implementing the PHY

To implement the PHY in a synthesized design outside of opt_design, enter:

```
implement_mig_cores
```

When you use this command, the Vivado tools implement the memory controller in the synthesized netlist without implementing the whole design. For more information, see the implement_mig_cores Tcl command in the *Vivado Design Suite Tcl Command Reference Guide* (UG835).

CAUTION! *Do not run the `implement_mig_cores` command more than once on an open design. Instead, close the design, reopen it, and run the command again.*

Copying I/O Port Assignments Between Designs

You can create a scoped XDC file to copy UltraScale architecture Memory IP I/O constraints from one design to another. This is useful if you want to:

- Perform memory I/O assignments in an IP example design and then copy the assignments to your design.
- Create an example design containing the Memory IP and related I/O assignments from your design.

***Note*:** Because top-level port names might differ between the designs, you must use a scoped XDC file to copy the constraints.

To create a scoped XDC file to copy Memory IP I/O constraints:

1. Open the elaborated or synthesized design to copy the I/O assignments from.
2. In the Tcl Console, enter the following command to write a scoped XDC file for the Memory IP I/O constraints:

```
write_xdc -cell <memory_ip_instance_name> -file <output file name>
```

3. Open the elaborated or synthesized design to copy the I/O assignments to.
4. In the Tcl Console, enter the following command to read the scoped XDC file for the Memory IP I/O constraints:

```
read_xdc -cell <memory_ip_instance_name> -file <output file name>
```

5. Select **File → Constraints → Save** to interpret the I/O constraints to the top-level ports and write the constraints into the target constraint file.

   ***Note*:** The Tcl command equivalent is `save_constraints`.

Chapter 5

High-Speed I/O Planning for Versal ACAP

The Vivado® tool flow for Versal® devices introduces the Advanced IO Wizard, an IP-based flow for creating and planning high speed SelectIO™ interfaces. For more information on SelectIO XPHY logic implementation using the Advanced IO Wizard refer to the Advanced IO Wizard Tutorial in *Advanced I/O Wizard LogiCORE IP Product Guide* (PG320). This section is an overview of how to access the Advanced IO planner using an Advanced IO Wizard example design.

The I/O Planning view includes the I/O Ports and Package Pins window. This tool only allows for nibble and bank-level granularity. If more granularity is required, use the classic pin planning tools. If the Versal architecture Advanced IO Wizard IP exists in the design, the banners of both tabs contain a message and a button to launch the Advanced I/O Planner. The Advanced I/O Planner understands any interface in the XPHY I/O block. If there is a hard or soft memory controller, it also appears in the Advanced I/O Planner. The placement of these interfaces is arranged based on a centralized hardware rule-based engine. There is a priority to this list such as hard memory controller, soft memory controller, followed by any SelectIO interface. You can use the Advanced I/O planner to either automatically or manually assign IP interface signal groups to specific nibble groups within the I/O banks. For more information on Advanced IO Planner, see the Using Advanced I/O Planner section of *Advanced I/O Wizard LogiCORE IP Product Guide* (PG320).

Using the Advanced IO Planner

To create a new IO planning project, complete the following steps:

1. Launch Vivado and create a new project targeting a Versal device as described in this step.
   - In the Quick Start section, select **Create Project**.
   - Click **Next**.
   - In the Project Name page, specify **Project name**, **Project location**, and check **Create project subdirectory**.
   - Click **Next**.
   - In the Project Type page, select **RTL Project** and **Do not specify sources a this time**.

- Click **Next**.
- In the Default Part page, choose a default part.
- Click **Next**.

  The New Project Summary page is displayed.
- Click **Finish**.

2. When the project setup is complete, in the Flow Navigator and click **IP Catalog** under the PROJECT MANAGER heading.
3. In the IP Catalog window, search Advanced IO Wizard IP and double-click it to open the Customize IP dialog box.
4. In the Customize IP dialog box, leave the IP configurations at default values and click **OK** to add the IP to the project.

Figure 26: **Customize IP Dialog Box**

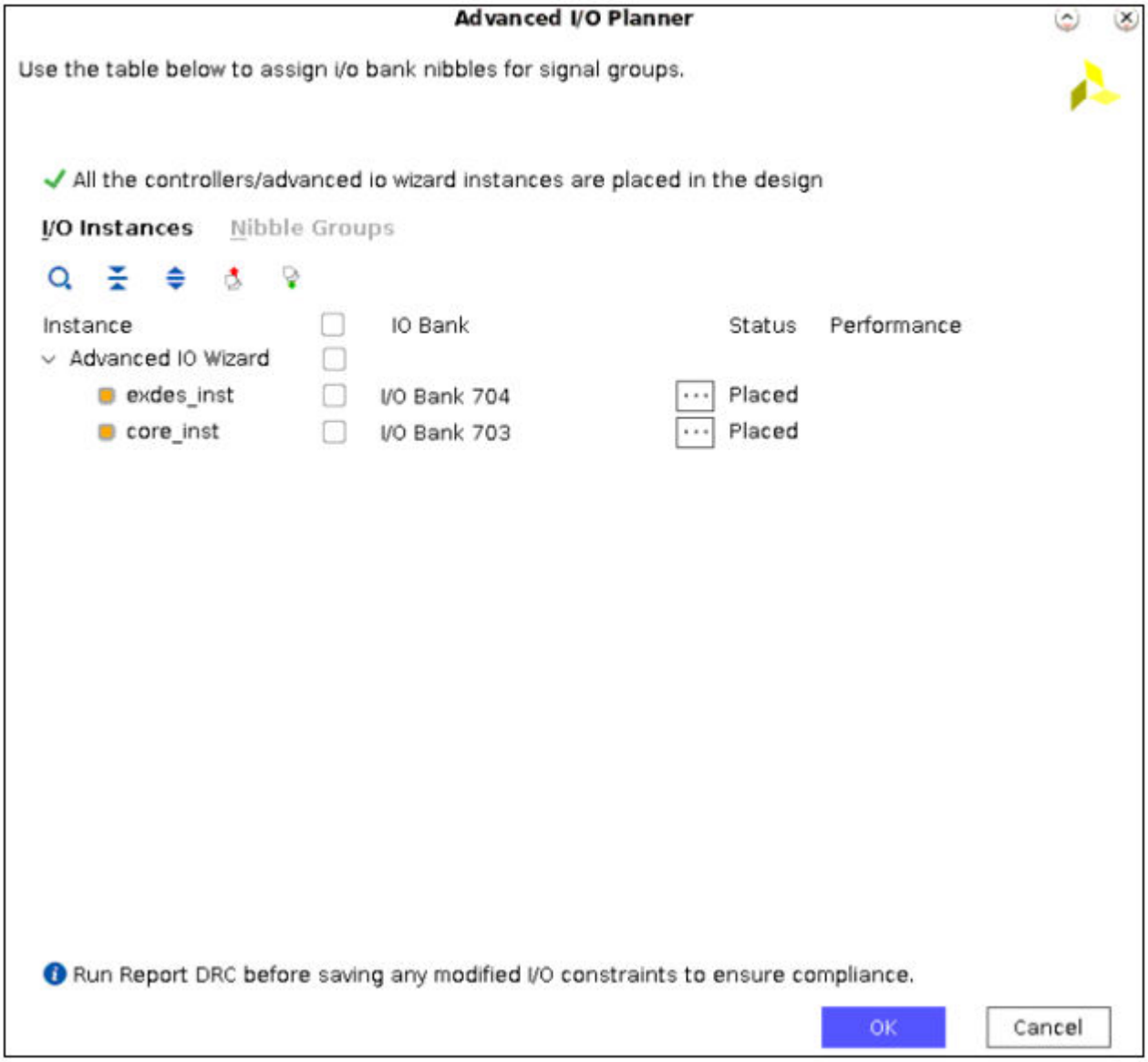
5. After the IP is added, a prompt appears for generating OOC products for the newly added IP. Click **Generate**.
6. Navigate to the Sources window and right-click the `advanced_io_wizard_0.xci` file and select the **Open IP Example Design** option.
7. Set up the example design by as shown in the following figure.

Figure 27: **Open IP Example Design Dialog Box**

8. Click **OK**.
9. In the Flow Navigator choose the **Run Synthesis** option under the SYNTHESIS heading.
10. After the synthesis completes, select **Open Synthesized Design** and click **OK**.
11. After opening the synthesized design, select **Window → I/O Ports**.

    The I/O Ports window is displayed.
12. In the I/O Ports window there is an option to open the Advanced I/O Planner tool as shown in the following figure. Click the link to configure port placement at bank level.

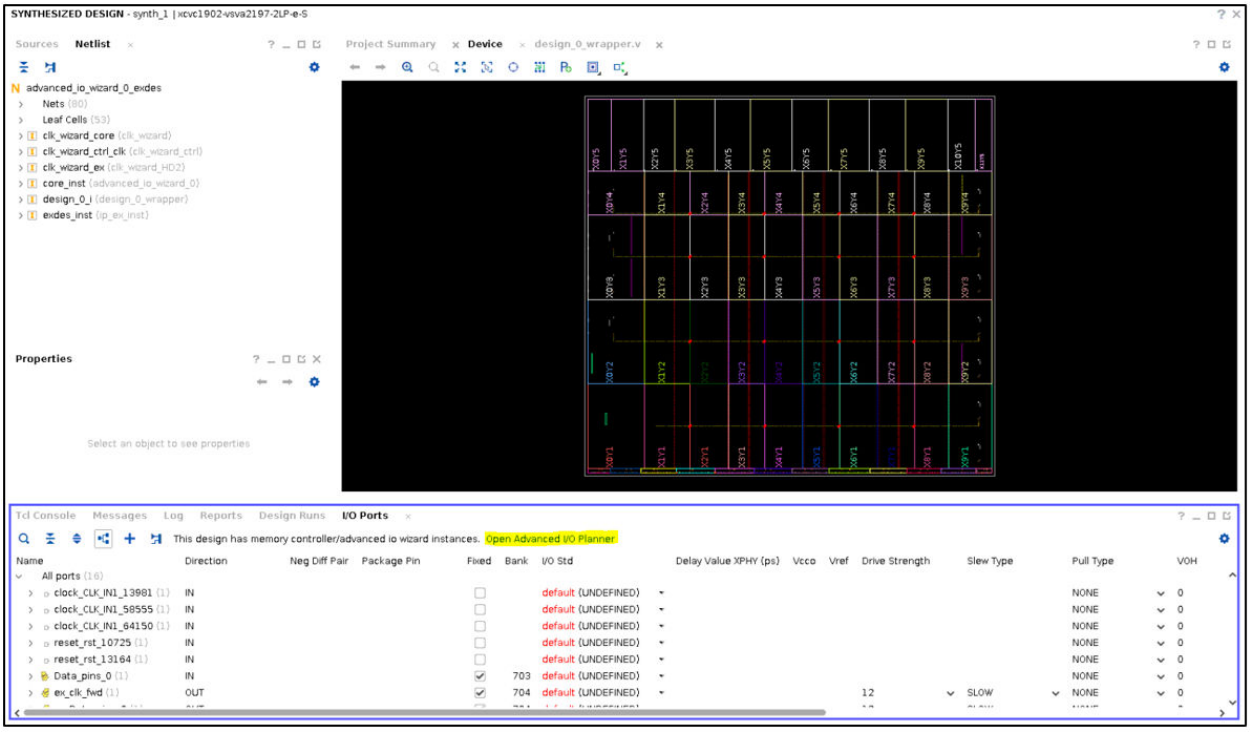
Figure 28: **Advanced I/O Planner**



13. Click **OK**.

    The following figure shows the example design.

Figure 29: **Example Design**



Chapter 6

Hard Block Planner

Hard Block Planning for Versal ACAP

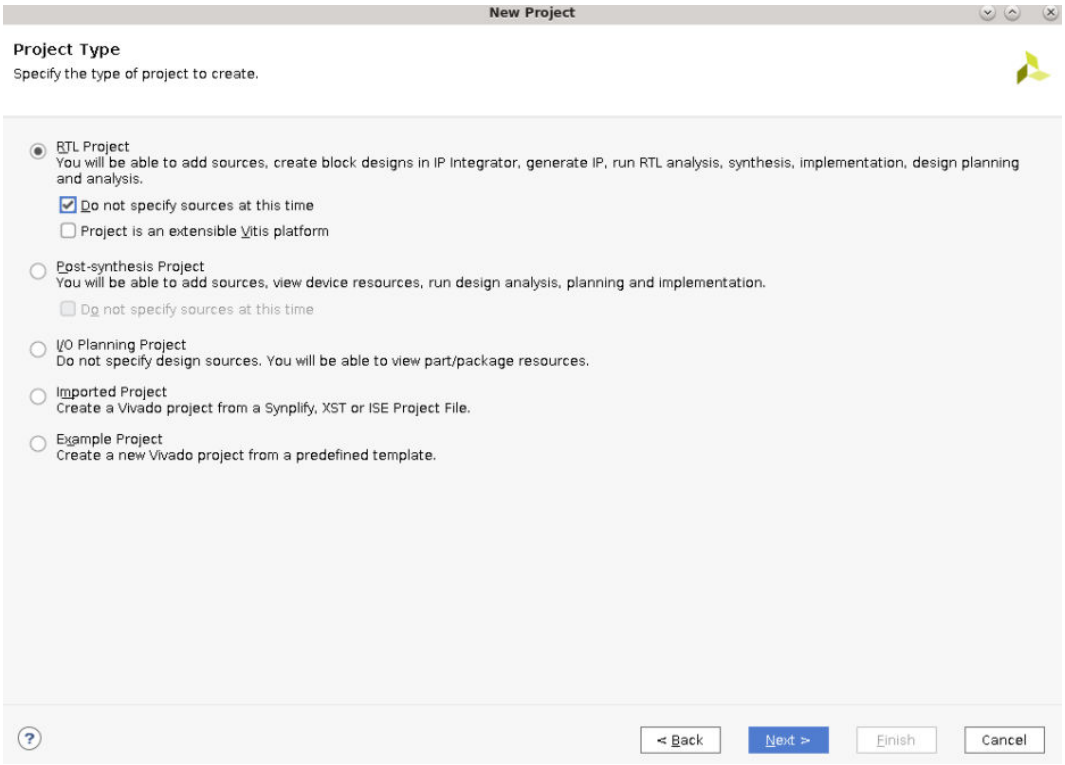
GT components are updated from Common/Channel to a GT_QUAD granularity for Versal ACAP. To enable some of the GT sharing use cases, GT wizard flows are modified to use the Vivado IP integrator. Use the Vivado IP integrator to build system designs that use single or multiple GT_QUADs. The design entry for custom IP that connects to GT_QUADs is through the Bridge IP, which instantiates, configures, and connects single or multiple GT Quad-based IP through Block Automation. Because GT_QUADs can be shared between multiple IPs, GT_QUAD and REFCLK locations are not assigned in IP Integrator.

The Hard Block Planner provides an intuitive user interface to assign GT_QUAD and REFCLK locations. The Hard Block Planner window groups GT_QUADs under Hard-IPs such as PCIe and DCMAC and provides visual feedback as to the location of the REFCLK with respect to the GT_QUADs. Furthermore, it provides an easy to use mechanism to assign GT_QUADs using the device sites. The Hard Block Planner provides visual feedback in the Device window for location of the REFCLK pins, the GT_QUADs and the Hard-IP blocks. Once you open a synthesized design, it reads and processes netlist objects and collects all hard-IPs available in a design. This planner allows you to cross-probe the location in Device window view for changing or assigning the Site. The Hard Block Planner option in the Windows menu appears only once you open a synthesized design or implemented design.

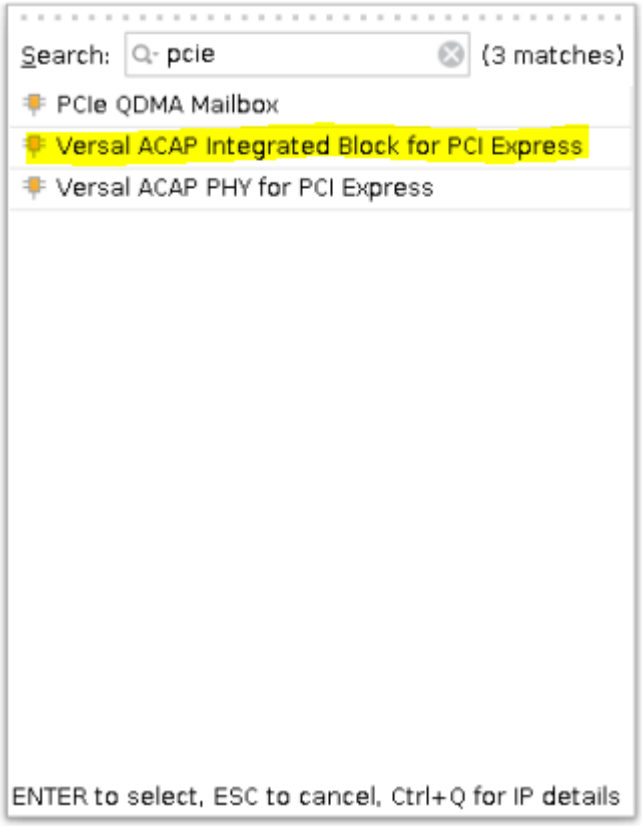
Using the Hard Block Planner

To create design for hard block planning, complete the following steps

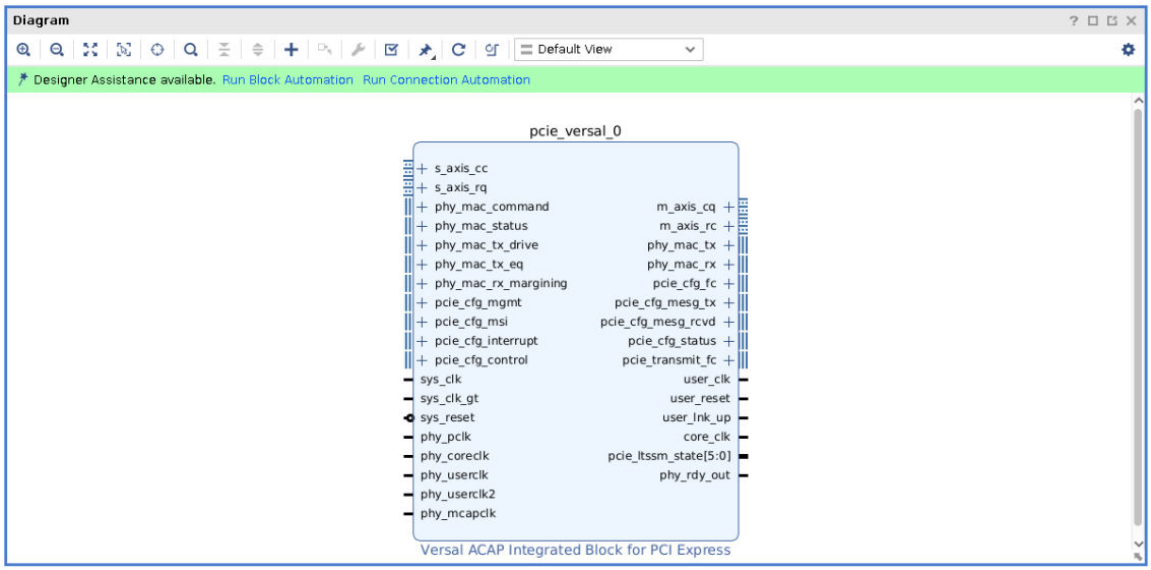
1. 1. Create a project that targets a Versal ACAP device. In this lab we will be creating a PCIE design that targets xcvc1902-vsvd1760-1LP-i-L. Ensure that the "**Do not specify sources at this time**" option is checked while creating an RTL project.



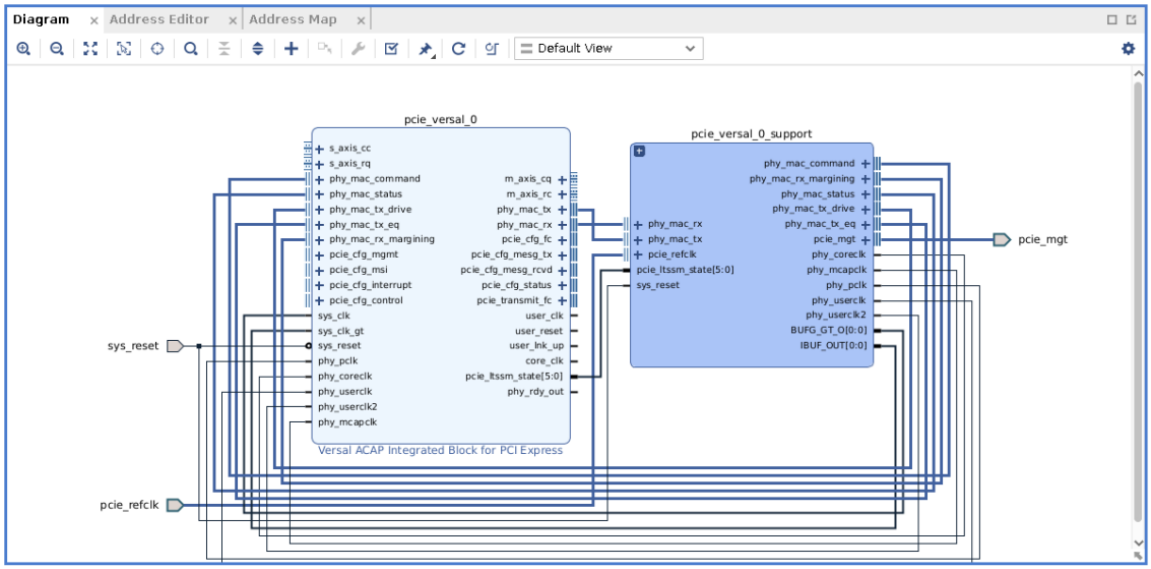
2. Once the project loads, click on the + button in the BD canvas to add Versal ACAP Integrated Block for PCI Express IP.



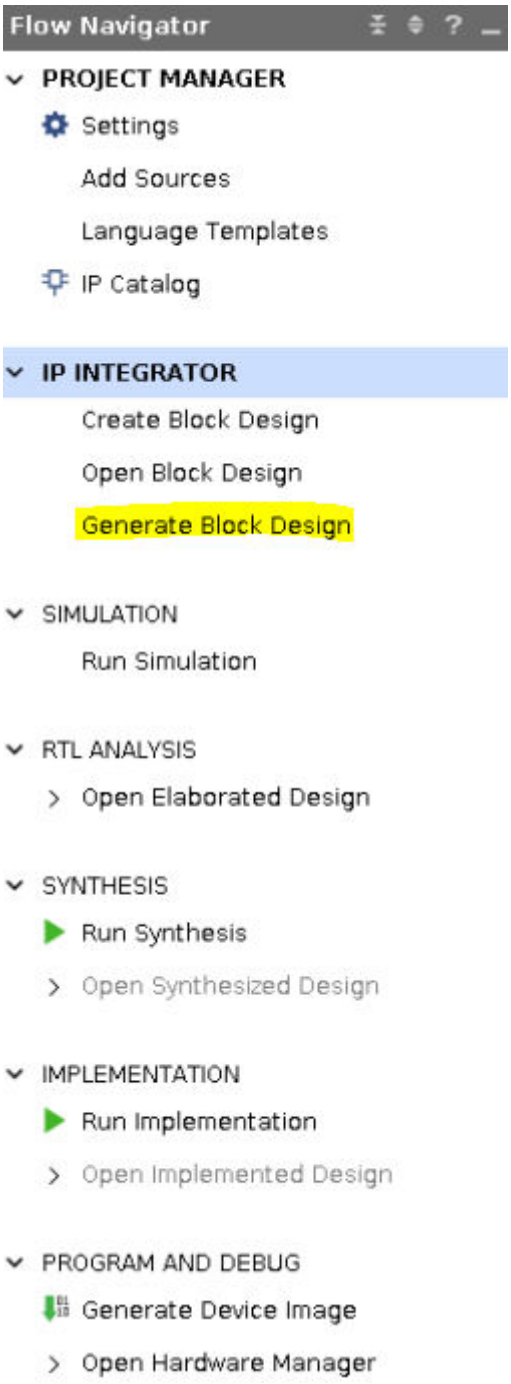
3. After adding the IP onto the BD canvas, click the **Run Block Automation** ribbon that appears above the BD canvas.

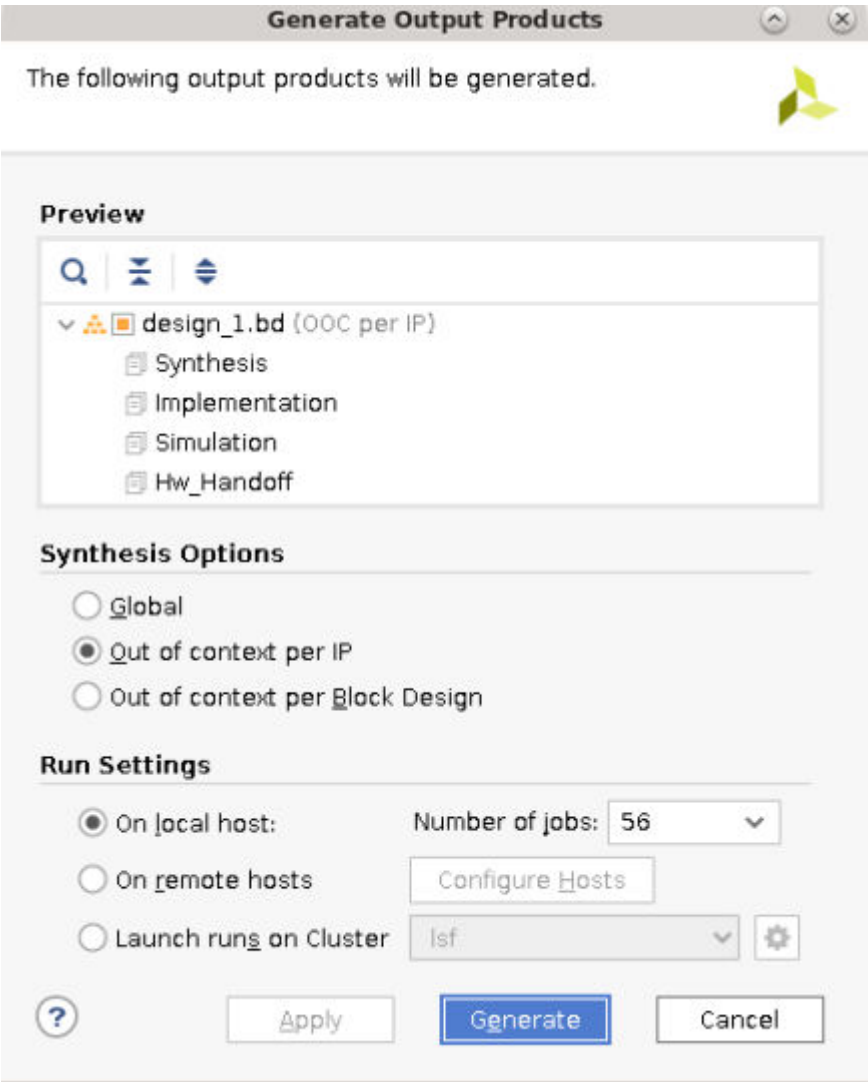


4. Running the block automation instantiates a pcie_versal_0_support block. This block contains the GT module that the PCIE requires to talk to the external world.

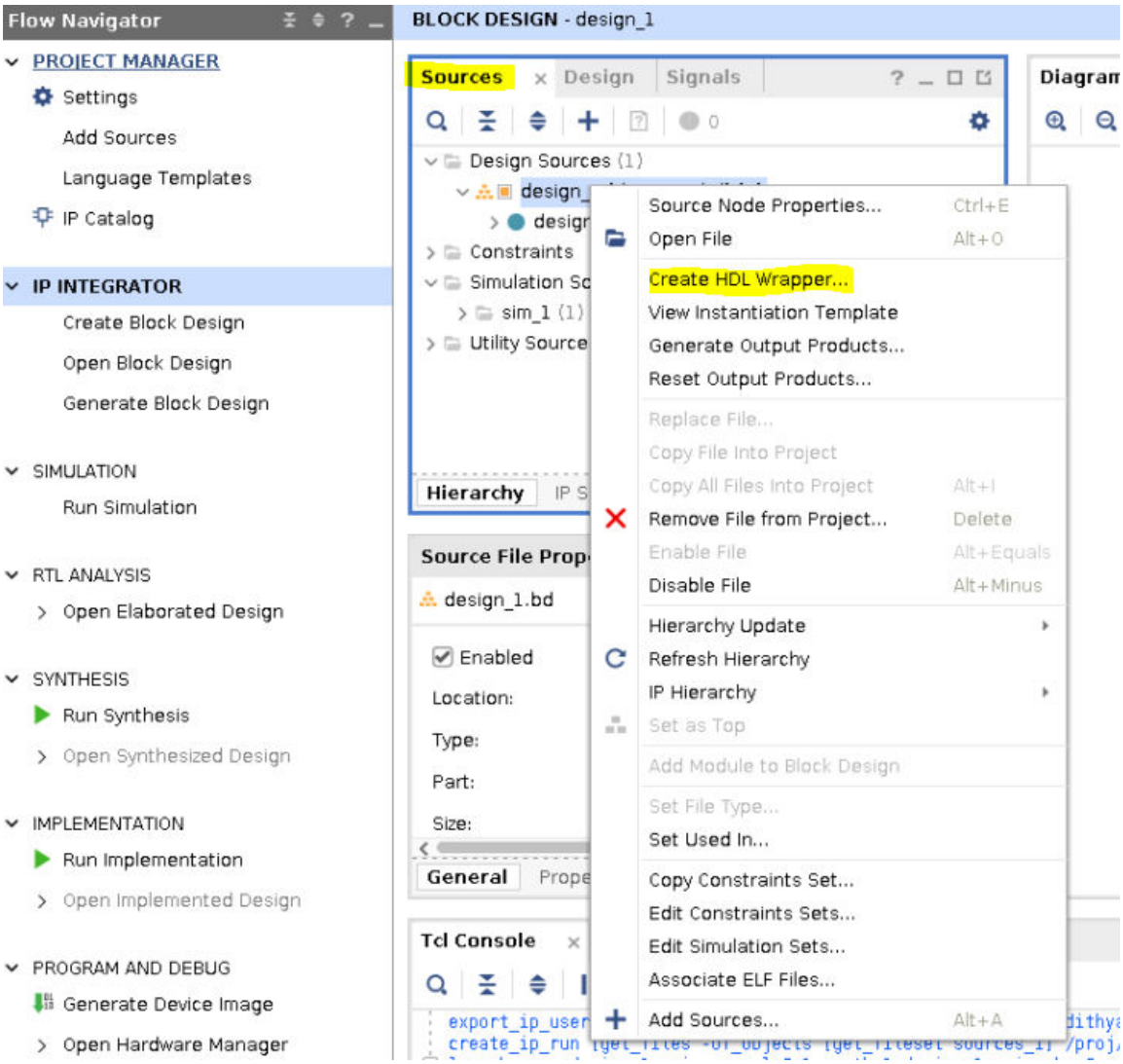


5. Once the block automation completes, click the **Generate Block Design** option under IP Integrator menu of the Flow Navigator window. When a subsequent prompt appears, leave the settings at default and click the **Generate** button.

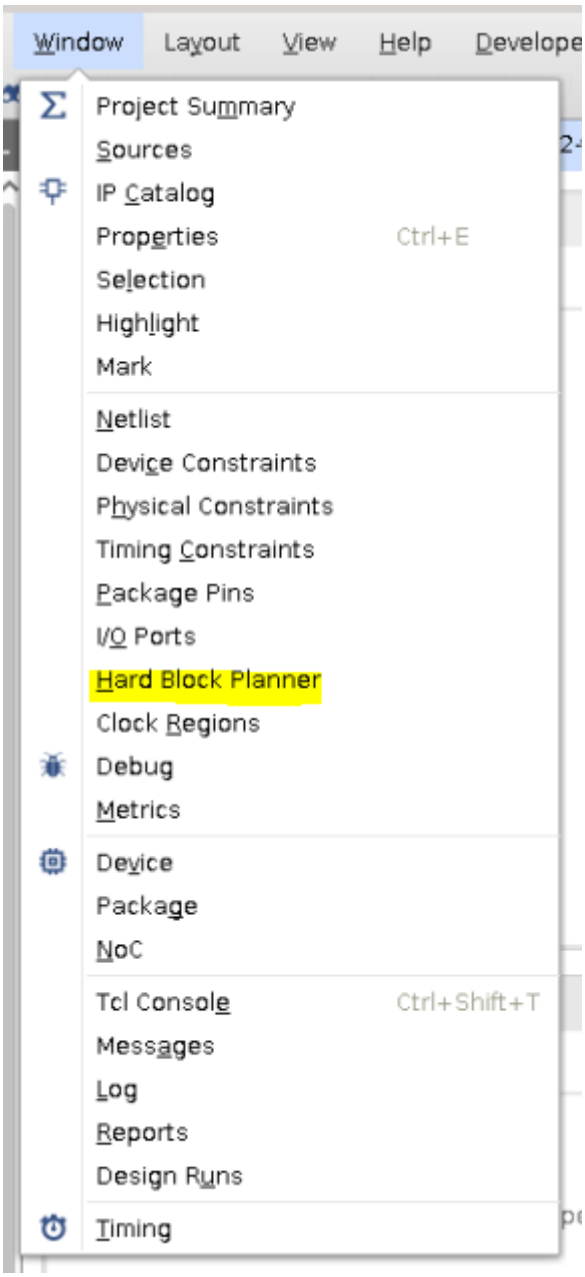

Flow Navigator
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Open Elaborated Design
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Design
PROGRAM AND DEBUG
Generate Device Image
Open Hardware Manager



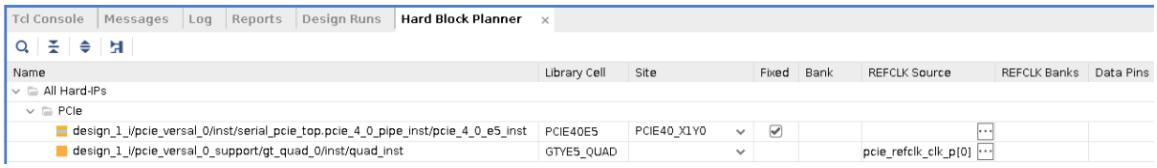
6. Create a top-level **HDL wrapper** before synthesizing the design.



7. The next step is to synthesize the design and open the synthesized design. Hard IP option in Windows menu appears only once you open synthesized design.

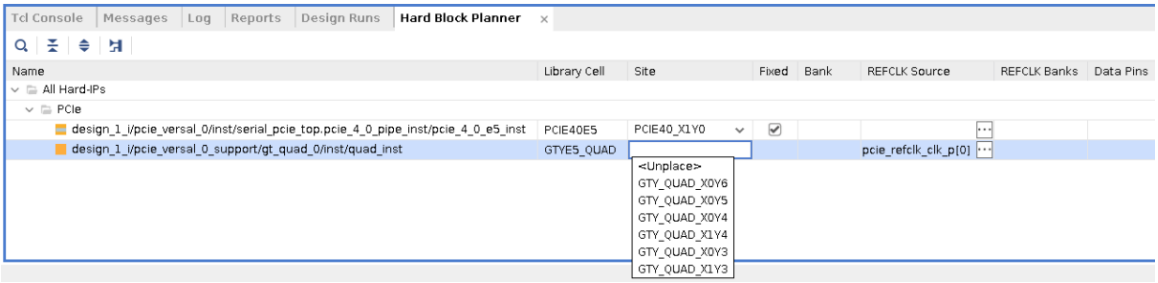


8. Once you open a synthesized design, it reads and processes netlist objects and collects all hard-IPs available in a design. This planner allows you to cross-probe the location in device window view for changing or assigning the Site.

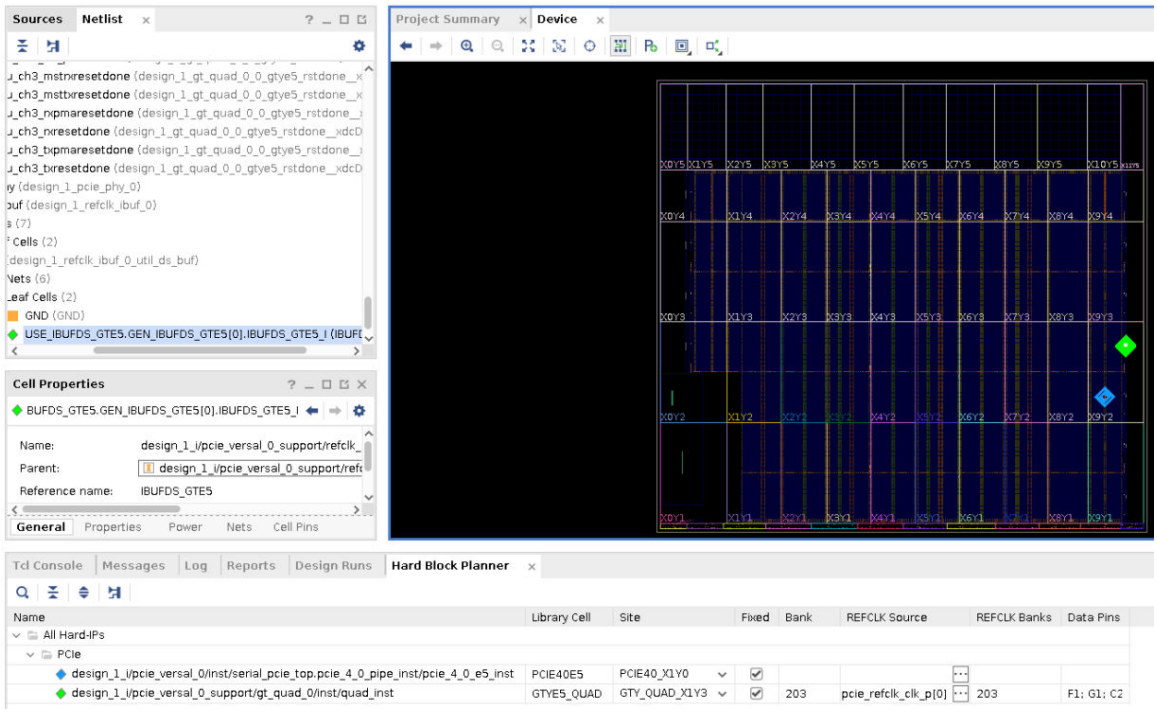


9. In our current design, we have 2 hard-IPs (PCI Express and GTY_QUAD) whose location can be planned using the Hard Block Planner tool. You can do so by selecting the drop-down menu under the Site column and picking a suitable site from the list.

Note: The site for the PCI Express hard IP is pre-determined by the PCIE Express IP in IPI.



10. For GT_QUADs, you can also plan the REFCLK source placement using the REFCLK Source column.



Chapter 7

Clock Planning

In clock planning, you determine how to use various clock resources on the Xilinx® device to distribute the clocks across the device. A Xilinx device is subdivided into columns and rows of clock regions. A clock region contains CLBs, DSP slices, block RAMs, interconnect, and associated clocking resources. The size and contents of a clock region vary by device type. For example, in UltraScale™ devices, the clock region spans 60 CLBs, 24 DSP slices, and 12 block RAMs with a horizontal clock spine (HCS) at its center. In 7 series devices, the clock region spans 50 CLBs and 1 I/O bank, which includes 50 I/Os, with a horizontal clock row (HROW) at its center.

A system clock, or board clock, is a primary clock that enters the design through an input port or a gigabit transceiver output pin. Each I/O bank contains clock-capable input pins to bring system clocks onto the device and into clock routing resources. In conjunction with dedicated clock buffers, the clock-capable input pins bring system clocks onto:

- Global clock lines
- I/O clocks lines within the same I/O bank and adjacent I/O banks
- Regional clock lines within the same clock region and vertically adjacent clock regions
- Clock management tiles (CMTs)

  Note: You can define the primary clock using the create_clock Tcl command. For more information on the `create_clock` command, see the *Vivado Design Suite Tcl Command Reference Guide* (UG835).

  When working with a synthesized or implemented design, you can manually place global and regional clock-related logic, such as BUFGCTRLs, MMCMs, BUFRs, and IDELAYCTRLs, using the Clock Resources window. You can also manually place clock logic in the Device window. Appropriate logic sites are displayed in the Device window for all device-specific resources. For more information on clock planning, see the one of the following Clocking Resources User Guide, depending on your device.

  . *7 Series FPGAs Clocking Resources User Guide* (UG472)

  . *UltraScale Architecture Clocking Resources User Guide* (UG572)

RECOMMENDED: *Xilinx recommends that you select clocking resources prior to pinout selection. This is because clocking selections can dictate a particular pinout and can also direct logic placement for that logic. Proper clocking selections can yield superior results.*

TIP: *The Vivado® tools handle clock planning automatically during implementation. You can then use interactive clock planning to manually address clocking issues.*

Locating Logic Cells

To locate logic cells for placing onto the device, do the following:

1. Select **Edit → Find**.
2. In the Find dialog box, specify **Cells** in the Find field, and define the criteria to locate the specific logic cell or cells.
3. From the Find Results window, drag logic cells onto the Clock Resources window or the Device window to assign to the appropriate device resource.

***Note*:** For more information, see this link in the *Vivado Design Suite User Guide: Using the Vivado IDE* (UG893).

TIP: *You can also locate physical resources on the device, such as global clock buffers, for placing logic cells. Specify **Sites** in the Find field, and define the criteria as needed. Select a result in the Find Results window to highlight the physical device resource in the Clock Resources window or Device window.*

Placing Clock Logic in the Device Window

To place clock logic manually:

1. In the Device window, zoom to locate the appropriate device site to place the logic.
2. Select the **Cell Drag & Drop Modes** toolbar button , and select **Create Site Constraint Mode** .
3. Select the logic cell to place from the Find Results, Schematic, Netlist, or I/O Ports window, and drag it onto the appropriate device resource in the Device window.

Chapter 8

Validating I/O and Clock Planning

After performing I/O and clock planning, you validate your design to ensure that it meets design requirements. The Vivado® tools allow you to run DRCs to check for violations and perform SSN Analysis to estimate switching noise levels. To perform a final validation on your I/O and clock assignments, you must implement the design and generate a bitstream.

Running DRCs

Running DRCs is one of the most critical steps in pin planning. DRCs check the current design against a specified set of design rule checks, known as a rule deck, and report any errors or violations. This section describes the required steps for running I/O port and clock-related DRCs and viewing DRC violations.

Note: For advanced DRC control, see this link in the *Vivado Design Suite User Guide: Using Tcl Scripting* (UG894). For information on `report_drc` and related Tcl commands, see the *Vivado Design Suite Tcl Command Reference Guide* (UG835).

Running I/O Port and Clock Logic Related DRCs

To select and run individual rules:

1. Select **Reports → Report DRC**.

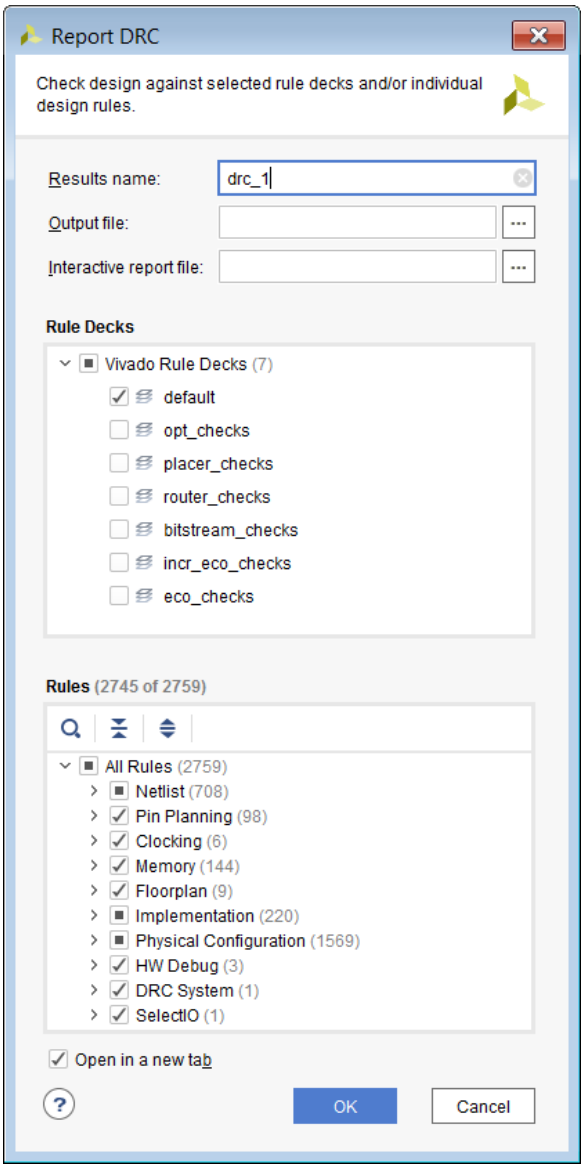
   Note: Alternatively, you can click **Report DRC** from the Flow Navigator, or enter the following command in the Tcl Console: `report_drc -name <results_name>`.

2. In the Report DRC dialog box (see the following figure), set the following options, and click **OK**:

   - **Results name:** Specify the name for the DRC results, which appear in a tab in the DRC window. Entering a unique name makes it easier to identify the results for a particular run during debug in the DRC window. By default, the output file name matches the name you enter.
   - **Output file:** Optionally, enter a file name to save the DRC results to a file. To select a path other than the default, use the browse button.

- **Rule decks:** Specify a rule deck to run on the design. A rule deck is a collection of design rule checks grouped for convenience. Design rule checks can exist in more than one rule deck. For example, the same rule can exist in both the opt_checks and placer_checks rule decks. The Vivado tools run the rule deck at the appropriate stage of the FPGA design flow, such as after synthesis or implementation.
  - **default:** Runs a default set of checks recommended by Xilinx
  - **opt_checks:** Runs checks associated with logic optimization.
  - **placer_checks:** Runs checks associated with placement.
  - **router_checks:** Runs checks associated with routing.
  - **bitstream_checks:** Runs checks associated with bitstream generation.
  - **timing_checks:** Runs checks associated with timing constraints.
  - **incr_eco_checks:** Checks validity of incremental ECO design modifications.
  - **eco_checks:** Runs checks on connectivity and placement after completing an engineering change order (ECO) that modifies the netlist. For example, this check is usually run on an implemented design when the connect_net Tcl command is used to modify the netlist.
- **Rules:** After specifying a rule deck, modify the rules to run as needed.

TIP: *If you want to run a non-standard set of design rule checks, you can create a custom rule deck using the* `create_drc_ruledeck` *and* `add_drc_checks` *Tcl commands as shown in Tcl Command Example for Defining a Custom Rule Deck. After you create the custom rule deck, it appears in the Rule Decks area of the Report DRC dialog box.*



Related Information

Tcl Command Example for Defining a Custom Rule Deck

Tcl Command Example for Running DRCs

```
report_drc -ruledecks default -file C:/Data/DRC_Rpt1.txt
```

Tcl Command Example for Defining a Custom Rule Deck

```
create_drc_ruledeck ruledeck_1
add_drc_checks -ruledeck ruledeck_1 [get_drc_checks {SYNTH-10 SYNTH-9
SYNTH-8 SYNTH-7 SYNTH-6 SYNTH-5 SYNTH-4}]
```

Note: For more information on the `create_drc_ruledeck` and related Tcl commands, see the *Vivado Design Suite Tcl Command Reference Guide* (UG835).

Enabling or Disabling Interactive DRCs

During I/O planning, the Vivado IDE runs basic checks to ensure a legal pinout. However, complete sign-off DRCs are only run during Vivado implementation. Therefore, you need to run your design through Vivado implementation to ensure final legal pinouts.

The interactive I/O placement routines check common error cases during pin placement. You can toggle this capability on and off with the Autocheck I/O Placement checkbox in the General tab of the Package window settings.

When you enable automatic checking, the tool does not allow placement of I/O ports on pins that cause a design issue. In Place I/O Ports Sequentially mode, if you attempt to place an I/O Port on a problematic pin, a tooltip appears that describes why the I/O port cannot be placed. The interactive DRCs are enabled by default.

IMPORTANT! *Many of these DRCs are only run on a synthesized or implemented design.*

The interactive I/O placement rules include:

- **Prohibiting:**
  - Placement on noise-sensitive pins associated with GTs or on I/O package pins that are potentially noise-sensitive
  - I/O standard violations
- **Ensuring:**
  - I/O standards are not used in banks that do not support them
  - Banks do not have incompatible VCC ports assigned
  - Banks that need VREF ports have free VREF pins
  - Proper assignment of global clocks and regional clocks (only with an imported netlist and XDC file)
  - Differential I/O ports are set to the proper sense pin
  - No output pins are placed on input-only pins

RECOMMENDED: *Xilinx recommends that you begin your I/O port placement with DRCs enabled. See the device documentation for more information on I/O ports and clock region specifications.*

Viewing DRC Violations

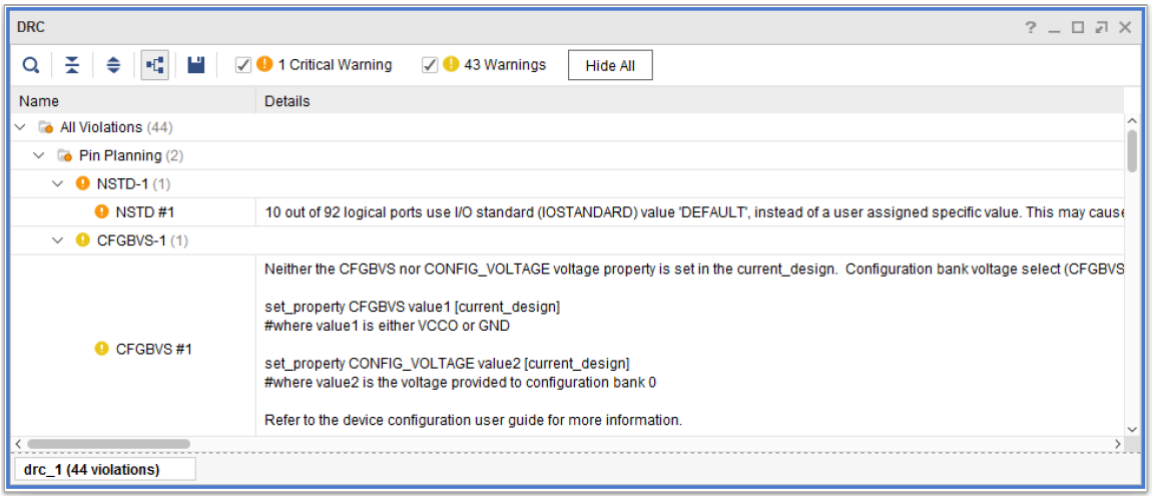
If violations are found, the DRC window opens, as shown in the following figure. The DRC window shows the rule violations, grouped under the various rule categories defined in the Report DRC dialog box. The rule violations are also categorized by severity and are color coded for quick review as follows:

- **Advisory:** Provides general status and feedback on design processing.
- **Warning:** Indicates that design results might be sub-optimal, because constraints or specifications might not be applied as intended.
- **Critical warning:** Indicates that certain user input or constraints will not be applied or do not adhere to best practices. Xilinx highly recommends that you examine these issues and make changes.

  ***Note*:** Critical warnings are promoted to errors during bitstream generation.
- **Error:** Indicates an issue that renders design results unusable and cannot be resolved without your intervention. The design flow stops.

TIP: *You can toggle the Hide Warnings and Informational Messages toolbar button to turn off warnings and informational messages to see only the errors reported.*

Figure 30: **DRC Violations**



Sorting DRC Violations

To sort DRC violations by severity, click the Severity column header as follows:

- Click the column header to sort in an increasing order.

- Click the column header again to sort in a decreasing order.

  Note: For more information, see the *Vivado Design Suite User Guide: Using the Vivado IDE* (UG893).

Viewing DRC Violation Properties

In the DRC window, right-click a violation message, and select Violations Properties. In the Violation Properties window, click the following views:

- **General:** Provides high-level information about the DRC rule violation, including type, severity, and description.
- **Details:** Provides information about the design elements that violate the rule. In addition, this view might include links to specific design objects that violate the DRC. Click these links to view the design object in the RTL Netlist window, Device window, Schematic window, or source RTL file.

Creating Custom DRCs

You can also use Tcl commands, such as `create_drc_check` and `create_drc_violation`, to create custom DRCs for use in the Vivado Design Suite. For more information, see this link in the *Vivado Design Suite User Guide: Using Tcl Scripting* (UG894). For more information on `create_drc_check` and related Tcl commands, see the *Vivado Design Suite Tcl Command Reference Guide* (UG835).

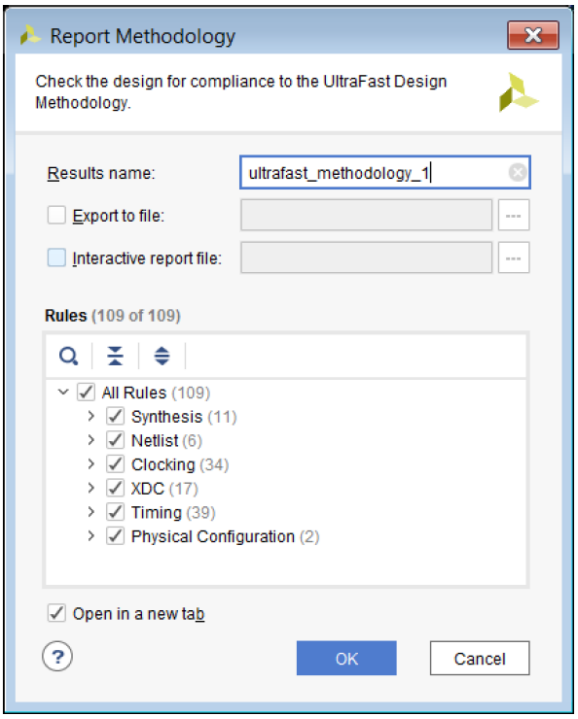
Querying DRCs

To get a list of the currently defined DRCs, use the `get_drc_checks` Tcl command. For more information, see `get_drc_checks` in the *Vivado Design Suite Tcl Command Reference Guide* (UG835), and see the *Vivado Design Suite User Guide: Using Tcl Scripting* (UG894).

Report Methodology

In 2016.1, some checks were moved from `report_drc` into the new `report_methodology` command. These checks include constraints and other rules that verify the design, such as logic mapping. The checks comply with the UltraFast™ Design Methodology and run a simplified rule check as shown in the Report Methodology dialog box.

Figure 31: **Report Methodology Dialog Box**



Run `report_methodology` the first time a design is synthesized and after a change in constraints, clocking topologies, or large logic changes to identify common design issues. For more information about running methodology checks, see this link in the *Vivado Design Suite User Guide: System-Level Design Entry* (UG895).

Working with SSN Analysis

The Vivado IDE provides analysis of the switching noise levels associated with the I/O of different devices. SSN analysis provides estimates of the disruption that simultaneously switching outputs can cause on other output ports in the I/O bank. SSN analysis incorporates I/O bank-specific electrical characteristics into the prediction to better model package effects on SSN.

I/Os are grouped into separate isolated I/O banks, and each I/O bank has unique power distribution networks and unique responses to switching activity. Because power distribution networks within a packaged FPGA have different responses to noise, it is important to understand the I/O standards and number of I/Os in a design as well as the response of the device power system to this switching.

Xilinx characterizes all banks through three-dimensional extraction and simulation. This information is incorporated into SSN analysis. SSN analysis uses the expected switching profile of a device to predict how the switching affects the power network of the system and in turn, how other outputs in the I/O bank are affected.

Note: SSN analysis analyzes output signals only, including the output of bidirectional ports, and ignores input signals in the calculation. As long as the I/O bank includes a sufficient margin, input and output levels are not affected.

IMPORTANT! *SSN analysis is the most accurate method available for predicting how output switching affects interface noise margins. The calculation and results are based on a range of variables. These estimates are intended to identify potential noise-related issues in your design and are not intended as final design sign-off criteria.*

Determining SSN Analysis Support

Not all devices support SSN analysis. To determine whether SSN analysis is available for the device targeted by the design open in memory, enter the following Tcl command:

```
get_property SSN_REPORT [get_property PART [current_design]]
```

To return a list of devices that support SSN analysis for the device family targeted by the design open in memory, enter the following Tcl command:

```
get_parts -filter "FAMILY == [get_property FAMILY [get_property PART \
[current_design]]] && SSN_REPORT"
```

Running SSN Analysis

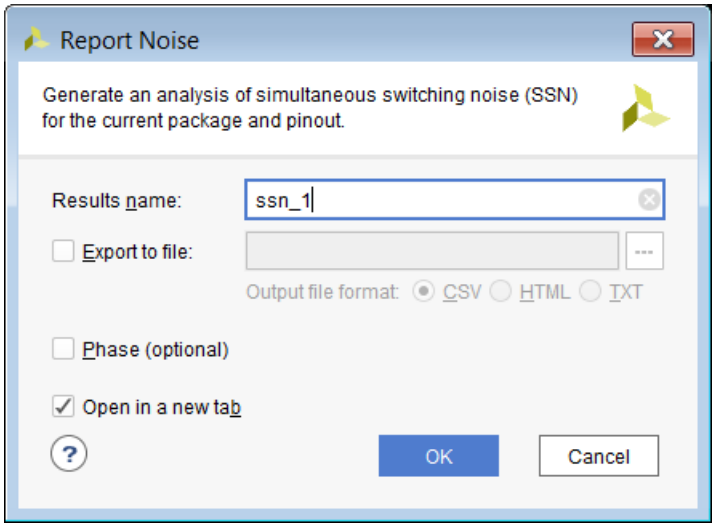
To run SSN analysis:

1. Select **Reports → IO → Report Noise**.

   Note: Alternatively, you can select **Report Noise** from the Flow Navigator.

2. In the Report Noise dialog box, set the following options, and click **OK**.
   - **Results Name:** Enter a name to identify the results in the Noise window.
   - **Export to File:** Exports the analysis to an external report file. Enter an output file name, or browse and select a location. Specify the output format for the file as either CSV or HTML.
   - **Phase:** Considers clocking information available in the design to more accurately report SSN noise. Clocks must be defined using the `create_clock` and `create_generated_clock` Tcl commands. The period, phase shift, and duty cycle of the generated clocks have significant impact on SSN analysis. For more information, see Adding Phase Information to SSN Analysis.

- **Open in a New Tab:** Opens the analysis results in a new tab in the Noise window. If you deselect this option, new results replace existing results in the Noise window.

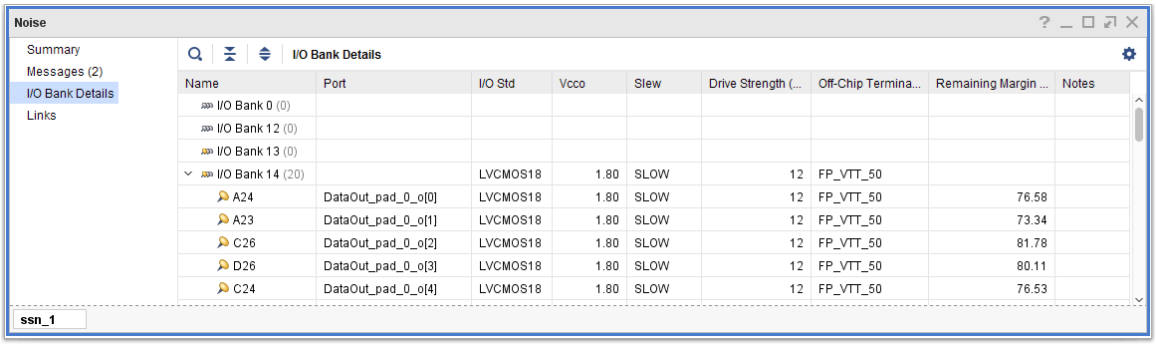


Related Information

Adding Phase Information to SSN Analysis

Viewing SSN Results

After the analysis is complete, the Noise window opens.



Name	Port	I/O Std	Vcco	Slew	Drive Strength (...	Off-Chip Termina...	Remaining Margin ...	Notes
I/O Bank 0 (0)								
I/O Bank 12 (0)								
I/O Bank 13 (0)								
I/O Bank 14 (20)		LVCMOS18	1.80	SLOW	12	FP_VTT_50		
A24	DataOut_pad_0_o[0]	LVCMOS18	1.80	SLOW	12	FP_VTT_50	76.58	
A23	DataOut_pad_0_o[1]	LVCMOS18	1.80	SLOW	12	FP_VTT_50	73.34	
C26	DataOut_pad_0_o[2]	LVCMOS18	1.80	SLOW	12	FP_VTT_50	81.78	
D26	DataOut_pad_0_o[3]	LVCMOS18	1.80	SLOW	12	FP_VTT_50	80.11	
C24	DataOut_pad_0_o[4]	LVCMOS18	1.80	SLOW	12	FP_VTT_50	76.53	

Click the links on the left side of the window to view different information about the SSN analysis. For example, click **I/O Bank Details** to view the following information:

- **Name:** Displays the I/O banks available in the device. Each I/O bank has pin icons indicating how full the bank is. A check mark indicates a passing result, and a red circle indicates a failure.
- **Port:** Displays the name of the user I/O in the FPGA design.
- **I/O Std, V_{CCO}, Slew, Drive Strength:** Displays the appropriate values for the port or bank.

- **Off-Chip Termination:** Displays the default terminations for each I/O standard, if one exists. Displays either None or a short description of the expected or defined off-chip termination style. For example, `FP_VTT_50` describes a far-end parallel 50 Ω termination to VTT termination style. `HSTL_1` describes a far-end 40 Ω termination to VTT. The full list of termination styles is available in one of the following user guides, depending on your device.
  - *7 Series FPGAs SelectIO Resources User Guide* (UG471)
  - *UltraScale Architecture SelectIO Resources User Guide* (UG571)

  For LVTTL (at 2 mA, 4 mA, 6 mA, and 8 mA) no termination is assumed. However, for LVTTL (at 12 mA and 16 mA) a far-end parallel termination of 50Ω to VTT is assumed. As a result of this termination, the available noise margin is less for signals with drive strength of 12 mA, or more, when compared to 2 mA to 8 mA. 7 series devices, Zynq®-7000, and UltraScale™ devices use this assumption for applicable drive strengths.

  To change the settings, use either of the following methods:

  - Use the CSV file import feature described in Importing a CSV File.
  - In the I/O Ports table, select an item from the drop-down list.
- **Remaining Margin %:** Displays the amount of noise margin that is left over after accounting for all SSN in the bank.
- **Notes:** Displays information about the I/O bank or groups.

IMPORTANT! *The SSN results are relative to the state of the design when the SSN analysis is run. It is not a dynamic report.*

Related Information

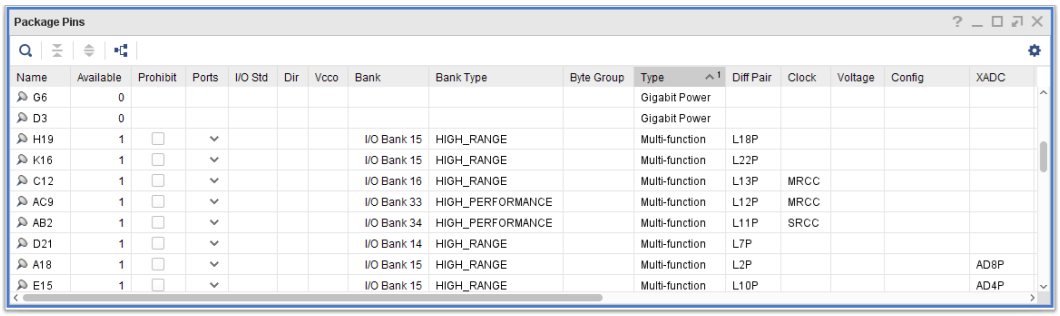
Importing a CSV File

Viewing I/O Bank Properties in SSN Results

You can select an I/O bank in the Noise window to display information about the I/O ports, pins, and groups assigned to the I/O bank in the I/O Bank Properties window. In the I/O Bank Properties window, you can:

- Select the **General** view to view information about the number and types of ports assigned to the I/O bank.
- Select the **Package Pins** or **I/O Ports** view to view the detailed information about the pins or ports within the bank, as shown in the following figure.

Figure 32: **I/O Bank Properties with Package Pins**



Package Pins

Name	Available	Prohibit	Ports	I/O Std	Dir	Vcco	Bank	Bank Type	Byte Group	Type	Diff Pair	Clock	Voltage	Config	XADC
G6	0									Gigabit Power					
D3	0									Gigabit Power					
H19	1						I/O Bank 15	HIGH_RANGE		Multi-function	L18P				
K16	1						I/O Bank 15	HIGH_RANGE		Multi-function	L22P				
C12	1						I/O Bank 16	HIGH_RANGE		Multi-function	L13P	MRCC			
AC9	1						I/O Bank 33	HIGH_PERFORMANCE		Multi-function	L12P	MRCC			
AB2	1						I/O Bank 34	HIGH_PERFORMANCE		Multi-function	L11P	SRCC			
D21	1						I/O Bank 14	HIGH_RANGE		Multi-function	L7P				
A18	1						I/O Bank 15	HIGH_RANGE		Multi-function	L2P				AD8P
E15	1						I/O Bank 15	HIGH_RANGE		Multi-function	L10P				AD4P

Improving SSN Results

To improve SSN results when a violation occurs:

- Use I/O standards that have a lower SSN impact for the failing group. Changing to a lower drive strength, a parallel-terminated DCI I/O standard, or a lower class of driver can improve SSN, for example, changing the SSTL Class II to an SSTL Class I.
- Spread the failing pins across multiple banks. This reduces the number of aggressive outputs on the power system of one bank.
- Spread the failing groups across multiple synchronous phases.
- Add phase information.

Adding Phase Information to SSN Analysis

You can increase margin in SSN analysis by adding phase information. By default, SSN analysis assumes that every output port toggles synchronously. This assumption covers the worst-case scenario that often yields an overly pessimistic SSN Analysis report. If clock information for the design is available, SSN analysis reports more accurate SSN noise.

To use this feature, enable SSN phase analysis using the following Tcl command:

```
report_ssn -phase
```

Enter clocking information using the `create_clock` and `create_generated_clock` Tcl commands. These commands provide the following required inputs to SSN analysis:

- Phase group

  Note: This groups generated clocks that belong to a single MMCM or PLL.
- Period
- Duty cycle
- Phase shift

Note: This covers an absolute phase shift from zero degrees.

For more information, see the *Vivado Design Suite Tcl Command Reference Guide* (UG835) and *Vivado Design Suite User Guide: Using Tcl Scripting* (UG894).

TIP: *When you enable SSN phase analysis, the SSN Analysis report shows information in the Phases column.*

Following are important considerations:

- Multiple master clocks do not improve SSN results. There must be multiple phases within each master clock to reduce SSN results.
- A single port within a phase group does not improve SSN results. For each clock group or phase group, there must be at least two ports.

To minimize SSN noise, shift the clock transitions of one clock with respect to another clock, for both clock signals in same phase-group as well as clock signals in different phase-groups. The magnitude (in ps) of these shifts are design and architecture dependent.

Following are additional considerations:

- For large designs, running SSN analysis with phase support might take in the tens of minutes.
- Shifting 180 degrees does not improve SSN results. Although clocking information includes rising and falling transition information, SSN analysis does not include actual output logic of the ports. When a clock transitions from Low to High, the output of a port can go in either direction. To ensure conservative SSN reporting, the algorithm assumes 180 degrees is the same as zero phase shift. Due to the lack of information on the output ports, the analysis overestimates SSN noise for ports with 180 degree shift. In reality, the SSN actual reduces with a 180 degree shift, but the algorithm cannot account for the reduction.
- Only 50% duty cycle is supported, and non-conforming clocks are considered as asynchronous signals.

Adding Temperature Information to SSN Analysis for 7 Series Devices

For 7 series devices and Zynq-7000, you can increase the accuracy of SSN analysis by specifying the temperature grade. You must use the correct temperature grade. To add temperature grade information, use one of the following Tcl commands, and then run SSN analysis.

```
set_operating_conditions -grade Commercial
set_operating_conditions -grade Industrial
set_operating_conditions -grade Military
set_operating_conditions -grade Q-Grade
set_operating_conditions -grade Extended
```

Note: For UltraScale architecture, temperature grade is included in the part name and automatically incorporated into the SSN analysis.

TIP: *To verify operating conditions. use the* `report_operating_conditions -grade` *Tcl command. To reset the temperature grade to the default, use the* `reset_operating_conditions -grade` *Tcl command.*

Following are additional considerations:

- By default, the temperature grade is based on the device used in the project.
- When running SSN analysis on an elaborated design, you cannot change the default temperature grade for the targeted part.
- Operating conditions are also used for power analysis. For information on operating conditions that impact power analysis, see this link in the *Vivado Design Suite User Guide: Power Analysis and Optimization* (UG907).

Chapter 9

Interfacing with the System Designer

As part of the iterative I/O and clock planning process, you can exchange information about the Xilinx® device pinout with the PCB or system designer by exporting the CSV files and IBIS models. Depending on PCB or design specification changes, you might need to reimport the pinout as described in Defining and Configuring I/O Ports. After completing the steps in the I/O and clock planning flow, you can return the pinout, along with device models for signal integrity analysis, using the CSV file and IBIS models.

Related Information

Defining and Configuring I/O Ports

Exporting I/O Pin and Package Data

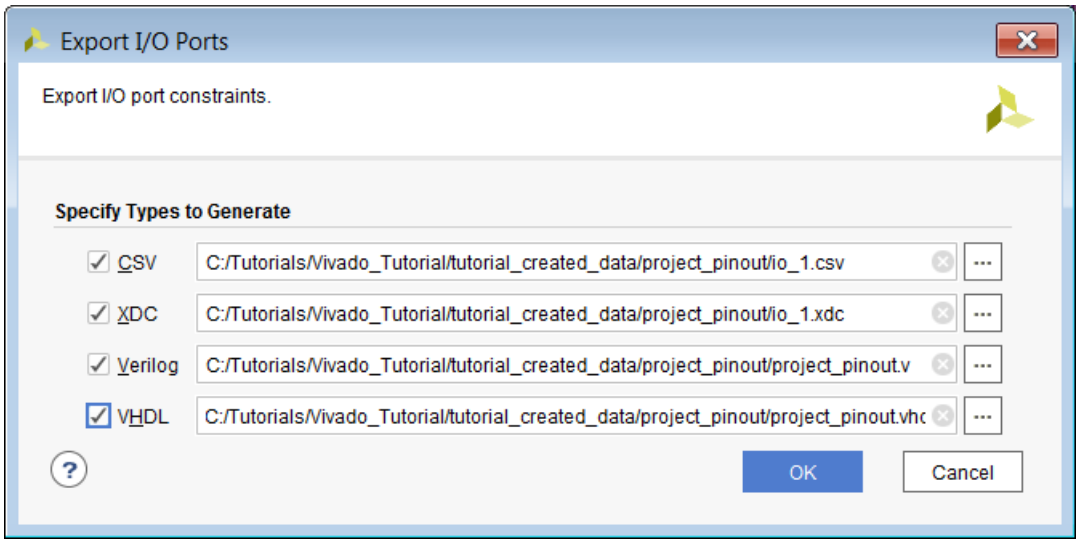
You can export I/O pin and package pin information for the following purposes:

- **I/O Pin Information:** You can export the I/O port list to a file for use in RTL coding or PCB schematic symbol creation.
- **Package Pin Information:** When working with an elaborated, synthesized, or implemented design, you can export the device package pin information to a CSV file. The package pin section of the exported list is a good starting point for defining I/O port definitions in a spreadsheet format. The exported information includes information about all of the package pins in the device as well as design-specific I/O port assignments and their configuration. Added columns and user-defined values are preserved and exported to the output file. For information on the exported CSV file format, see Defining and Configuring I/O Ports and Using I/O Port Lists in CSV File Format.

To export the I/O ports list information:

1. Select **File → Export → Export I/O Ports**.
2. In the Export I/O Ports dialog box, specify the type of I/O port to generate and the path, and click **OK**.

   Note: Only fixed constraints, which indicate that the ports are user-assigned, appear in the XDC file.



Related Information

Defining and Configuring I/O Ports
Using I/O Port Lists in CSV File Format

Generating IBIS Models

The Input/Output Buffer Information Specification (IBIS) is a device modeling standard. IBIS allows for the development of behavioral models used to describe the signal behavior of device interconnects. These models preserve proprietary circuit information, unlike structural models such as those generated from Simulation Program with Integrated Circuit Emphasis (SPICE) simulations. IBIS buffer models are based on V/I curve data produced either by measurement or by circuit simulation.

IBIS models are constructed for each IOB standard, and an IBIS file is a collection of IBIS models for all I/O standards in the device. An IBIS file also contains a list of the used pins on a device that are bonded to IOBs configured to support a particular I/O standard, which associates the pins with a particular IBIS buffer model.

The IBIS standard specifies the format of the output information file, which contains a file header section and a component description section. The Golden Parser was developed by the IBIS Open Forum Group (www.ibis.org) to validate the resulting IBIS model file by verifying that the syntax conforms to the IBIS data format.

When you export an IBIS model in the Vivado® IDE, the tool outputs a `.ibs` file. This file comprises a list of pins used by your design, the signals internal to the device that connect to those pins, and the IBIS buffer models for the IOBs connected to the pins.

Exporting IBIS Models

To better understand the signal integrity at the system level, PCB designers often need to simulate the design with IBIS models. Designers must consider signal integrity issues such as crosstalk, ground bounce, and SSN. IBIS models help characterize current-voltage (I-V) curves and parasitic information of the packaged device.

TIP: *You can download generic IBIS models from the Xilinx Downloads page.*

From the Vivado IDE, you can generate IBIS models from the design and per-pin package data. The Vivado IDE uses the netlist and implementation details from the design, and combines that information with the available per-pin parasitic package information to create a custom IBIS model for the design.

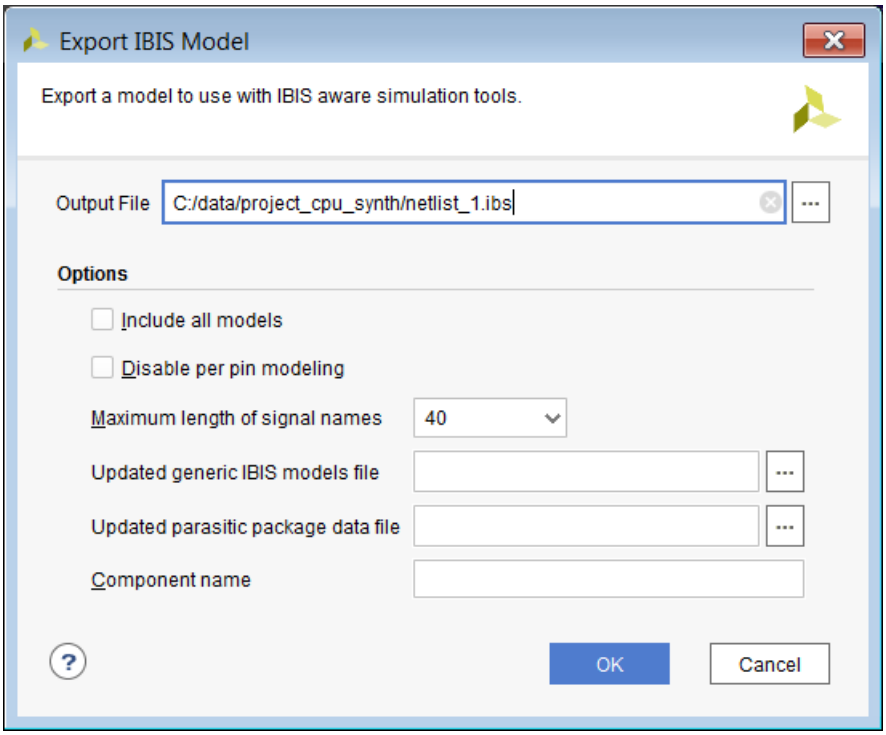
With an elaborated, synthesized, or implemented design open, you can export an IBIS file for use in analyzing the design as follows:

1. Select **File → Export → Export IBIS Model**.
2. In the Export IBIS Model dialog box, set the following options, and click **OK**:
   - **Output File:** Specify the file name and path for the output IBIS file.
   - **Include all models:** Include all available I/O buffer models for this device. By default, only buffer models used in the design are included.
   - **Disable per pin modeling:** Disable inclusion of the per-pin modeling of the package. This is the path from the die pad of the device to the pin of the package. With per-pin modeling disabled, the package is reduced to a single RLC transmission line model applied to all pins and defined in the [Package] section of the IBIS file.
   - **Maximum length of signal names:** Truncate signal names to the specified limit
     - **40:** Truncate signal names to 40 characters, which is supported by IBIS version 4.2 as the default.
     - **20:** Truncate signal names to 20 characters.
     - **Unlimited:** Do not truncate signal names.
   - **Updated generic IBIS model file:** Optionally, provide an IBIS model file for the device. This is used to override the IBIS models found in the installation under the parts directory.

     IMPORTANT! *The IBIS model file is required for devices that do not have IBIS models included with software installation.*
   - **Updated parasitic package data file:** Optionally, provide a parasitic package file (`.pkg` extension) file to use for per-pin extractions. This is used to override the parasitic package file found in the installation hierarchy under the parts directory.

     IMPORTANT! *The parasitic package file is required for devices that do not have IBIS models included with software installation.*

- **Component Name:** Optionally, specify a new component name to change the default value, which is the device family.



Interfacing with the PCB Design

The Vivado pin planner provides you with an effective way to select the pin assignments for the design. Choosing correct resources enables a faster and cleaner design process. The recommendations in this section help prevent board layout, pin assignment, and FPGA resource conflict. For more information on PCB and pin planning, see one of the following guides for your device:

- *7 Series FPGAs PCB Design Guide* (UG483)
- *UltraScale Architecture PCB Design User Guide* (UG583)
- *Zynq-7000 SoC PCB Design Guide* (UG933)

Part Compatibility

Set the part compatibility for your design as described in Defining Alternate Compatible Parts. The Vivado IDE identifies the pins that are common to all selected alternate parts and assigns prohibit constraints to pins that are not common to all devices, eliminating the possibility of using these pins. For more information, see the product table for the targeted device.

Related Information

Defining Alternate Compatible Parts

DRC

To check the overall integrity of the I/O assignments, run DRCs as described in Running DRCs. It is important that you resolve all warnings and errors reported by the DRCs before starting board planning.

Related Information

Running DRCs

SSN Analysis

To generate an estimate of potential noise disruptions, run SSN analysis as described in Working with SSN Analysis. Xilinx recommends that you address noise-related issues before starting board planning. For more information, see the SelectIO™ resources for your device:

- *7 Series FPGAs SelectIO Resources User Guide* (UG471)
- *UltraScale Architecture SelectIO Resources User Guide* (UG571)

and the Memory Resources resources for your device:

- *UltraScale Architecture Memory Resources User Guide* (UG573)
- *Zynq-7000 SoC and 7 series Devices Memory Interface Solutions* (UG586)

IBIS Simulation

To run IBIS simulations, use the IBIS file generated by the Vivado IDE as described in Generating IBIS Models.

Related Information

Generating IBIS Models

Package Trace Length

Instead of giving trace length data, Vivado Design Suite provides trace delay data, which is the most accurate way to estimate package delay.

1. Open a design in Vivado, either RTL, netlist or implemented.

2. Select **File → Export → Export I/O Ports** for a CSV type spreadsheet that shows the min and max package delays for each pin. The min/max trace delays are also displayed in the Package Pins window for every package pin within two separate columns.
3. The following Tcl commands can be used if there is no project:

```
link_design -part <part_number>
write_csv <file_name>
```

For example:

```
link_design -part xc7k410tffg900-2
write_csv flight_time
```

CSV Export

After running DRCs and SSN analysis, export the CSV file to assist with board planning as described in Exporting I/O Pin and Package Data.

Related Information

Exporting I/O Pin and Package Data

Supported Third-Party PCB Tools

To optimize the I/O assignments within the context of the entire board, Xilinx also supports Cadence Allegro FPGA System Planner and Mentor Graphics I/O Designer. For more information, see the third-party tool documentation.

Appendix A

Using I/O Port Lists in CSV File Format

CSV File

A CSV file is a standard file format used by FPGA and board designers to exchange information about device pins and pinout. For more information, see Importing a CSV File and Exporting I/O Pin and Package Data.

Following are descriptions of the CSV columns. For more information on each property, see the *Vivado Design Suite Properties Reference Guide* (UG912).

- **I/O Bank:** Specifies the I/O Bank in which the pin is located. The tool fills in this field for all pins in the device. Values are a number or blank. This is not required in the input CSV file.
- **Pin Number:** Specifies the name (or location) of the package pin. The tool writes this out for all pins in the device. This is not required in the input file. If used for input, it is used to define placement. Values are legal pins in the device.
- **Site:** Specifies an alternate part name for the package pin. This field is specified by the tool and is unused if specified in the input CSV file.

  Note: Prior to 2016.1, this column was called IOB Alias.
- **Site Type:** Specifies the pin name from the device data sheet. This field is specified by the tool and is unused if specified in the input CSV file.
- **Min/Max Trace Delay (ps):** Specifies the delay between the pad site of the die and the ball on the package, in picoseconds. This is specified by the tool to help the board engineer match trace delays. The Trace Delay fields are in the output file only. They are not expected in the input file.
- **Trace Length (um):** Specifies the length of the internal trace between the package pin and the die pad.

  Note: This is not published for most devices. Use trace delay instead and see Package Trace Length for more information.

- **Prohibit:** Specifies a prohibit site. Certain sites can be prohibited to prevent user I/O from being added to the site. For example, sites can be prohibited to:
  - Prohibit ease board layout issues.
  - Reduce crosstalk between signals.
  - Ensure that a pinout works between multiple FPGAs in the same package.

    Note: In the XDC file, this is represented by a PROHIBIT property.
- **Interface:** An optional user-specified grouping for an arbitrary set of user I/O. For example, this field provides a means to specify a relationship for the data, address, and enable signals for a memory interface. Values are a text string or blank.
- **Signal Name:** The name of the user I/O in the FPGA design. Values are a string or blank for an unassigned Package Pin.
- **Direction:** The direction of the signal. Values are IN, OUT, INOUT, or blank when a user I/O is not assigned to the site.
- **DiffPair Type:** This value instructs the software about which pin is the N side of a differential pair, and which pin is the P side. This is used for differential signals only. The tool uses this column instead of a naming convention to determine which pin is the N side of the pair, and which pin is the P side. Values are P, N, or blank when a user I/O is not assigned to the site.
- **DiffPair Signal:** Specifies the name of the other pin in the differential pair. Values are the name of the user I/O or blank when unused.
- **IO Standard:** Specifies the I/O standard for a specific user I/O. When this field is blank for a user I/O, the tool uses the appropriate device defaults. Values are a legal I/O standard for the user I/O in the device or blank.
- **Drive:** Drive strength of the I/O standard for a specific user I/O. Not all I/O standards accept a drive strength. If this field is blank, the tool uses the default. Values are a number or blank.
- **Slew Rate:** Specifies the I/O standard slew rate for a specific user I/O. Not all I/O standards accept a slew rate. If this field is blank, the tool uses the default. Values are FAST, MEDIUM (UltraScale™ Architecture only), and SLOW.
- **OUTPUT_IMPEDANCE:** (Supported by all architectures after 7 series) Specifies the driver impedance for HSTL, SSTL, HSUL, LVDCI, HSLVDCI, and POD drivers to match the characteristic impedance of the driven line. The OUTPUT_IMPEDANCE attribute defines the value of source termination at the driver for both DCI and non-DCI versions of the supported standards.
- **PRE_EMPHASIS:** (Supported by all architectures after 7 series) Allows pre-emphasis for certain I/O standards to improve signal integrity of high-frequency signals by reducing intersymbol interference and minimizing the effects of transmission line losses.

- **LVDS_PRE_EMPHASIS:** (Supported by all architectures after 7 series) Allows pre-emphasis for LVDS I/O standards to improve signal integrity of high-frequency signals by reducing intersymbol interference and minimizing the effects of transmission line losses.
- **Pull Type:** Specifies the pull type for the selected port. When using 3-state output (OBUFT) or bidirectional (IOBUF) buffers, the output can have a weak pull-up resistor, a weak pull-down resistor, or a weak "keeper" circuit. For input (IBUF) buffers, the input can have either a weak pull-up resistor or a weak pull-down resistor.
- **IN_TERM:** (7 series devices only) Defines the optional IN_TERM or OUT_TERM driver impedance properties. This is most commonly left blank but is supported for production devices. Using this terminal definition overrides the SLEW and DRIVE STRENGTH properties and is not supported in SSN calculations.
- **DQS_BIAS:** (Supported by all architectures after 7 series) Defines an optional DC bias at the inputs of certain pseudo-differential and true differential IO standards.
- **DIFF_TERM:** Turns the built-in differential termination on or off.
- **OFFCHIP_TERM:** Specifies the external board level termination of the I/O. This is used for SSN calculations. If the field is left blank, the tool uses the expected terminations in the SSN calculations and shows this expected termination by default in the SSN report and I/O Ports table.

  ***Note*:** For information on the expected terminations as well as the corresponding shortened names that display in the tool, see the SelectIO™ Resources User Guide for your device.

  - *7 Series FPGAs SelectIO Resources User Guide* (UG471)
  - *UltraScale Architecture SelectIO Resources User Guide* (UG571)
- **Board Signal:** Specifies the name of the signal coming into the I/O from the board-level design.
- **Board Voltage:** Specifies the voltage level of the signal coming into the I/O from the board-level design.
- **ODT:** (Supported by all architectures after 7 series) Reports the design's optional on-chip Termination.

IMPORTANT! *The columns listed above are read in as constraint values and any additional columns will be retained as user defined columns in the Package Pins window. Additional constraints for I/O should be imported through XDC.*

Related Information

Importing a CSV File
Exporting I/O Pin and Package Data
Package Trace Length

Differential Pairs in the CSV File

There are several properties used to define differential pairs in the CSV file

- Signal Name
- DiffPair Signal
- DiffPair Type
- I/O Standard

The other values in the CSV file are used to validate a diff pair, to ensure they are fully compatible, but they are not used to define the pair. You can define differential pairs in the CSV file in the following ways:

- **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 properties such as I/O Standard are compatible when forming the diff pair.
- **Single-Link 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 creates the differential pair if all other properties 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 file. The tool creates 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 differential pair I/O standards (for example, DIFF_HSTL, DIFF_SSTL) and Signal Names that infer the N and P side. The tool infers a differential pair if all other properties are compatible.

Appendix B

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Xilinx Support.

Solution Centers

See the Xilinx Solution Centers for support on devices, software tools, and intellectual property at all stages of the design cycle. Topics include design assistance, advisories, and troubleshooting tips.

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.

References

These documents provide supplemental material useful with this guide:

1. *UltraFast Design Methodology Guide for Xilinx FPGAs and SoCs* (UG949)
2. *Vivado Design Suite User Guide: Design Flows Overview* (UG892)
3. *Vivado Design Suite User Guide: System-Level Design Entry* (UG895)
4. *Clocking Wizard LogiCORE IP Product Guide* (PG065)
5. *Vivado Design Suite User Guide: Designing with IP* (UG896)
6. *Vivado Design Suite User Guide: Synthesis* (UG901)
7. *Vivado Design Suite User Guide: Implementation* (UG904)
8. *Vivado Design Suite Tcl Command Reference Guide* (UG835)
9. *Vivado Design Suite Properties Reference Guide* (UG912)
10. *7 Series FPGAs Configuration User Guide* (UG470)
11. *UltraScale Architecture Configuration User Guide* (UG570)
12. *Zynq UltraScale+ Device Technical Reference Manual* (UG1085)
13. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
14. *7 Series FPGAs SelectIO Resources User Guide* (UG471)
15. *UltraScale Architecture SelectIO Resources User Guide* (UG571)
16. *Vivado Design Suite User Guide: Using the Vivado IDE* (UG893)
17. *7 Series FPGAs Packaging and Pinout Product Specification* (UG475)
18. *UltraScale and UltraScale+ FPGAs Packaging and Pinouts Product Specification* (UG575)
19. *Zynq-7000 SoC Packaging and Pinout Product Specifications* (UG865)
20. *Zynq-7000 SoC Technical Reference Manual* (UG585)
21. *7 Series FPGAs GTX/GTH Transceivers User Guide* (UG476)
22. *UltraScale Architecture GTH Transceivers User Guide* (UG576)
23. *UltraScale Architecture Memory Resources User Guide* (UG573)
24. *Zynq-7000 SoC and 7 series Devices Memory Interface Solutions* (UG586)
25. *UltraScale Architecture-Based FPGAs Memory IP LogiCORE IP Product Guide* (PG150)

26. *7 Series FPGAs Clocking Resources User Guide* (UG472)
27. *UltraScale Architecture Clocking Resources User Guide* (UG572)
28. *Vivado Design Suite User Guide: Using Tcl Scripting* (UG894)
29. *Vivado Design Suite User Guide: Power Analysis and Optimization* (UG907)
30. *7 Series FPGAs PCB Design Guide* (UG483)
31. *UltraScale Architecture PCB Design User Guide* (UG583)
32. *Zynq-7000 SoC PCB Design Guide* (UG933)
33. *Zynq UltraScale+ Device Technical Reference Manual* (UG1085)
34. *Versal ACAP Clocking Resources Architecture Manual* (AM003)
35. *Versal ACAP SelectIO Resources Architecture Manual* (AM010)
36. IBIS Open Forum Group (www.ibis.org)
37. Xilinx Downloads
38. Vivado Design Suite Documentation

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:

1. Designing FPGAs Using the Vivado Design Suite 1 Training Course
2. Designing FPGAs Using the Vivado Design Suite 2 Training Course
3. Designing FPGAs Using the Vivado Design Suite 3 Training Course
4. Designing FPGAs Using the Vivado Design Suite 4 Training Course
5. Vivado Design Suite QuickTake Video: Designing with UltraScale Device Memory IP
6. Vivado Design Suite QuickTake Video Tutorials

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

© Copyright 2012-2021 Xilinx, Inc. Xilinx, the Xilinx logo, Alveo, Artix, Kintex, Kria, Spartan, Versal, Vitis, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.