# Adder/Subtracter v12.0

## *LogiCORE IP Product Guide*

**Vivado Design Suite**

**PG120 February 4, 2021**

XILINX

# Table of Contents

## Appendix B: Debugging



## Appendix C: Additional Resources and Legal Notices

# Introduction

The Xilinx LogiCORE™ IP Adder/Subtracter core provides LUT and single DSP slice add/sub implementations. The Adder/Subtracter module can create adders (A+B), subtracters (A–B), and dynamically configurable adder/subtracters that operate on signed or unsigned data. The function can be implemented in a single DSP slice or LUTs (but currently not a hybrid of both). The module can be pipelined.

# Features

- Generates adder, subtracter and adder/subtracter functions
- Supports two's complement-signed and unsigned operations
- Supports fabric implementation inputs ranging from 1 to 256 bits wide
- Supports DSP slice implementations with inputs up to 58 bits
- Optional carry input and output
- Optional clock enable and synchronous clear
- Optional bypass (load) capability
- Option to set the B Value to a constant
- Optional pipelined operation

| LogiCORE IP Facts Table | |
|---|---|
| **Core Specifics** | |
| Supported Device Family(1) | Versal™ ACAP<br>UltraScale+™ Families<br>UltraScale™ Architecture<br>Zynq®-7000 SoC<br>7 Series |
| Supported User Interfaces | N/A |
| Resources | Performance and Resource Utilization web page |
| **Provided with Core** | |
| Design Files | Encrypted RTL |
| Example Design | Not Provided |
| Test Bench | Not Provided |
| Constraints File | N/A |
| Simulation Model | Encrypted VHDL |
| Supported S/W Driver | N/A |
| **Tested Design Flows(2)** | |
| Design Entry | Vivado® Design Suite<br>System Generator for DSP<br>Vivado |
| Simulation | For supported simulators, see the Xilinx Design Tools: Release Notes Guide. |
| Synthesis | Vivado Synthesis |
| **Support** | |
| Release Notes and Known Issues | Master Answer Record: 54491 |
| All Vivado IP Change Logs | Master Vivado IP Change Logs: 72775 |
| Xilinx Support web page | |

**Notes:**

1. For a complete listing of supported devices, see the Vivado IP catalog.
2. For the supported versions of third-party tools, see the Xilinx Design Tools: Release Notes Guide.

# Overview

## 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. This document covers the following design processes:

- **Hardware, IP, and Platform Development:** Creating the PL IP blocks for the hardware platform, creating PL kernels, subsystem functional simulation, and evaluating the Vivado timing, resource and power closure. Also involves developing the hardware platform for system integration. Topics in this document that apply to this design process include:
  - Port Descriptions
  - Clocking
  - Resets
  - Customizing and Generating the Core

## Feature Summary

The Adder/Subtracter core implements area-efficient, high-performance adders, subtracters and adder-subtracters. The core can be customized to use either FPGA logic or a DSP slice to construct the adder.

## Applications

The Adder/Subtracter can be used to implement general purpose fixed-point arithmetic, or as a building block for custom accumulator or counter logic for a wide range of applications, such as address generation.

# Licensing and Ordering

This Xilinx LogiCORE™ IP module is provided at no additional cost with the Xilinx® Vivado® Design Suite under the terms of the Xilinx End User License. Information about other Xilinx LogiCORE IP modules is available at the Xilinx Intellectual Property page. For information on pricing and availability of other Xilinx LogiCORE IP modules and tools, contact your local Xilinx sales representative.

# Product Specification

## Resource Utilization

For details about resource utilization, visit the Performance and Resource Utilization web page.

## Performance

For details about performance, visit the Performance and Resource Utilization web page.

## Port Descriptions

Signal names for the Adder/Subtracter core are shown in Figure 2-1 and described in Table 2-1. Figure 2-1 shows the SSET and SINIT pins which appear only on fabric implementations.

**IMPORTANT:** *The DSP slice implementations do not support SSET and SINIT.*

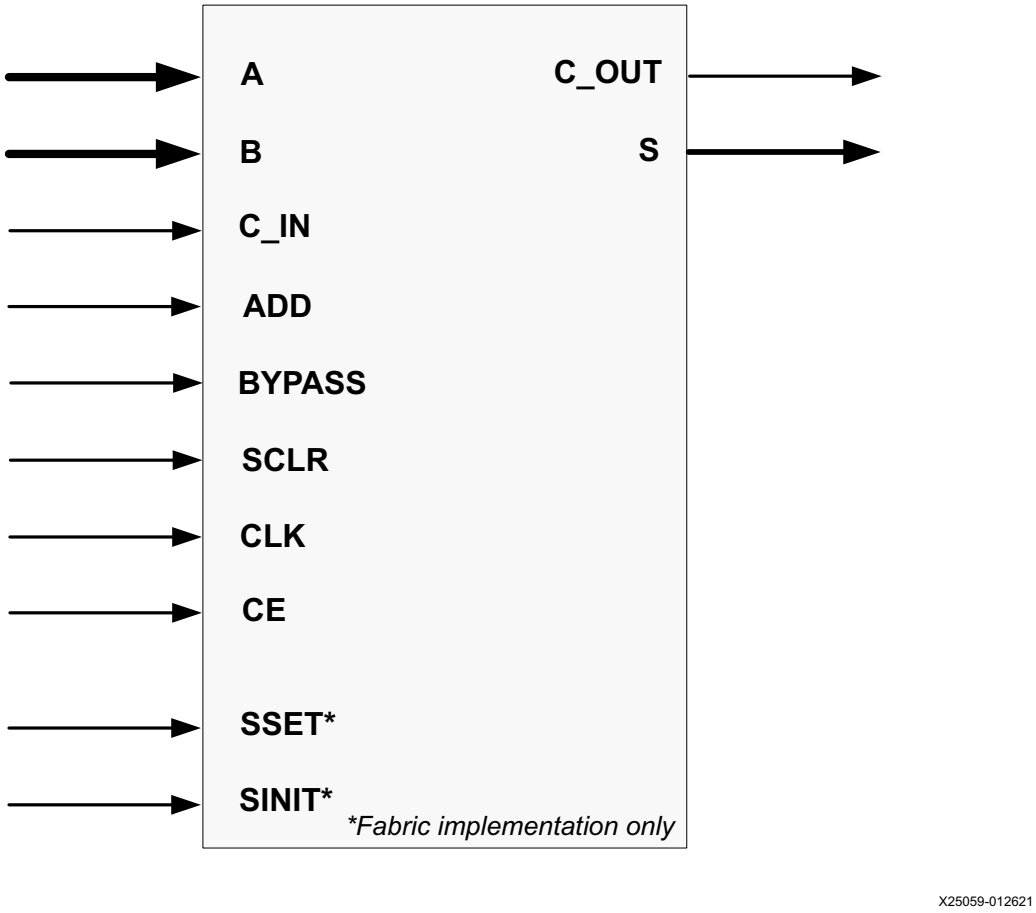


*Figure 2-1:* **Core Symbol**

*Table 2-1:* **Core Signal Pinout**

| Name | Direction | Description |
|---|---|---|
| A[N:0] | Input | A Input bus |
| B[M:0](1) | Input | B Input bus |
| ADD | Input | Controls the operation performed by an Adder/Subtracter (High = Addition, Low = Subtraction) |
| C_IN | Input | Carry Input |
| C_OUT | Output | Carry Output |
| S[P:0] | Output | Output bus |
| BYPASS | Input | Bypass control signal loads B port onto S port |
| CE | Input | Active-High Clock Enable |
| CLK | Input | Clock signal: rising edge |
| SCLR | Input | Synchronous Clear: forces outputs to a Low state when driven High |
| SINIT(2) | Input | Synchronous Initialization - forces outputs to a user defined state when driven High |

*Table 2-1:* **Core Signal Pinout *(Cont'd)***

| Name | Direction | Description |
|---|---|---|
| SSET(2) | Input | Synchronous Set - forces outputs to a High state when driven High |

**Notes:**

1. B port is not present if **Constant Input** = TRUE and **Bypass** = FALSE. A user-defined core-internal constant is applied in place of the B operand.
2. Available for fabric implementations only.

# Designing with the Core

This chapter includes guidelines and additional information to facilitate designing with the Adder/Subtracter core.

## General Design Guidelines

### Output Widths

Caution must be exercised when choosing output widths to accommodate input widths and sign types, as described in Table 3-1. The error trapping of inadequate output width is disabled to allow the System Generator tool to zero pad or sign-extend inputs and to allow input and output widths to be equal. Warnings are given when an inadequate output width is chosen but the Vivado® IP catalog does not error. It assumes that the appropriate padding has been added to allow for bit growth and only issues a warning.

*Table 3-1:* **Availability of Carry Outputs and Output Data Type/Size vs. Input Data Type**

| Input types | Output Type | Conditions | Valid Output Widths | Carry Out |
|---|---|---|---|---|
| Both Unsigned | Unsigned | Add Mode = Add, Subtract | Q[(1)] | Available |
| | | | Q + 1 | Not Available |
| | | Add Mode = Add_Subtract | Q + 1 | Available |
| | | | | |
| One Unsigned, One Signed | Signed | Signed input wider | Q,<br>Q + 1 | Not Available |
| | | Inputs equal width or unsigned input wider | Q + 1,<br>Q + 2 | Not Available |
| Both Signed | Signed | None | Q | Not Available |
| | | | Q + 1 | Not Available |

**Notes:**

1. Q = Max(A Input Width, B Input Width).

## Pipelined Operation

The Adder/Subtracter module can be optionally pipelined to improve speed. The pipelined operation is controlled by the latency parameters. Set **Latency Configuration** to Automatic to achieve optimal pipelining for maximum speed. Set **Latency Configuration** to Manual to allow a valid number of pipeline stages to be entered in the **Latency** parameter.

### *DSP Slice*

For DSP slice implementations, the single DSP slice can be pipelined with 0, 1, or 2 stages of registers. **Latency Configuration** = Automatic optimizes the latency for speed. For **Latency** = 1, only output registers are present. For **Latency** = 2, output and input registers are present.

### *Fabric Implementations*

For fabric implementations, pipelining is achieved by splitting the input buses into many bus-slices (that equals the number of pipelining stages), and doing as much work as possible on each bus-slice in the first stage; adding them together and storing the results and the carry-out of each. In the second stage, the carry-out from the least-significant slice is then fed into the next-higher result, which produces a carry-out that is fed into the next result in the following stage, until the carry has propagated to the top.

Because less data needs to be stored, this is more efficient than the more intuitive technique which stores the inputs for each slice until the carry-in for that slice is generated. Additionally, the design is smaller and more easily routed.

After power-up or reset, the pipelined module takes several clock cycles, specified by the latency control, for the outputs to become valid.

If bypass is requested on a pipelined module, the bypass value appears on the outputs after the number of clock cycles specified by the latency control. Note that if both bypass and clock enable are requested, bypass priority must be set so that bypass does not override clock enable. For pipelined modules, the resource usage is roughly **Latency** times bigger than the non-pipelined equivalent.

***Note:*** Pipelining results in a significant increase in area usage in order to increase clock speed. If latency is required but area is more important than speed, add an SRL16-based shift register to the S output of this module for optimal area usage. See the RAM-based Shift Register core for this functionality.

## Clocking

The core requires a single clock, CLK, and is active-High triggered.

If selected, the active-High clock enable, CE, stalls all core processes when deasserted.

## Resets

The core has a single, active-High synchronous reset, SCLR. Asserting SCLR for a single cycle resets all registers in the core.

The priority of SCLR and CE pins can be selected when customizing the core.

# Design Flow Steps

This chapter describes customizing and generating the core, constraining the core, and the simulation, synthesis and implementation steps that are specific to this IP core. More detailed information about the standard Vivado® design flows and the IP integrator can be found in the following Vivado Design Suite user guides:

- *Vivado Design Suite User Guide: Designing IP Subsystems using IP integrator* (UG994) [Ref 1]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 2]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 3]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 5]

## Customizing and Generating the Core

This chapter includes information about using the Vivado® Design Suite and System Generator for DSP to customize and generate the Adder/Subtracter core.

If you are customizing and generating the core in the Vivado IP integrator, see the *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 1] for detailed information. IP integrator might auto-compute certain configuration values when validating or generating the design. To check whether the values do change, see the description of the parameter in this chapter. To view the parameter value you can run the **validate_bd_design** command in the Tcl Console.

You can customize the IP for use in your design by specifying values for the various parameters associated with the IP core using the following steps:

1. Select the IP from the IP catalog.
2. Double-click the selected IP or select the Customize IP command from the toolbar or right-click menu.

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 2] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 3].

## Core Parameters

The Adder/Subtracter Vivado IDE provides fields to set the parameters values for the required instantiation. This section provides a description of each field.

- **Implement using**: Sets the implementation type to Fabric or DSP slice.
- **A Input Width**: Sets the width of the port `A` input. In IP integrator, this parameter is auto-updated.

  ***Note:*** Mixed signed/unsigned input types can require a bit growth of 2 bits on the output in some cases. See Table 3-1, page 10.
- **A Input Type**: Sets the type of the port `A` data to Signed or Unsigned. In IP integrator, this parameter is auto-updated.
- **B Input Width**: Sets the width of the port `B` input. In IP integrator, this parameter is auto-updated.

  ***Note:*** Mixed signed/unsigned input types can require a bit growth of 2 bits on the output in some cases. See Table 3-1, page 10.
- **B Input Type**: Sets the type of the port `B` data to Signed or Unsigned. In IP integrator, this parameter is auto-updated.
- **Constant Input** and **Constant Value**: When **Constant Input** is TRUE, port `B` is set to the value that is specified with the parameter **Constant Value**. **Constant Value** must be entered in binary format and must not exceed **B Input Width**. In most cases specifying port `B` to be a constant creates a module without port `B`. The only exception to this is when bypass functionality is requested, as port `B` is needed to provide the bypass data in this case. The default setting is for the port `B` value to be provided by port `B`.
- **Output Width**: Sets the output width. The valid range varies depending on the settings of **A Input Width**, **A Input Type**, **B Input Width**, and **B Input Type**, as shown in Table 3-1, page 10. See Output Widths in Chapter 3 for more information about sufficiency and warning messages.
- **Add Mode**: Sets the mode of operation of the module. Valid values are Add, Subtract, and Add/Subtract. If an adder/subtracter is specified, the `ADD` pin sets the mode of operation.
- **Carry In**: When this parameter is set to TRUE, a `C_IN` port is created. This is an active-High, carry-in port for adders and a programmable (active-High/active-Low with **Borrow In/Out Sense**) carry-in port for subtracters and adder/subtracters in subtract mode.
- **Carry Out**: When set to TRUE, this parameter creates port `C_OUT` which is the synchronous active-High carry-out from the adder and adder/subtracter in add mode and the programmable (active-High/active-Low with **Borrow In/Out Sense**) borrow-out from the subtracter or adder/subtracter in subtract mode. See Table 3-1, page 10 for information about when these outputs are permitted.

- **Bypass**: When set to TRUE, creates a BYPASS pin. Activating the BYPASS pin sets the output to be the value given on port B. This functionality is used for creating loadable counters and accumulators.
- **Bypass and Clock Enable (CE) Priority**: This parameter controls whether or not the BYPASS input is qualified by **Clock Enable**. When set to Bypass_Overrides_CE, the activation of the BYPASS signal also enables the register. When set to CE_Overrides_Bypass, the register must have CE active to load the B port data.
- **Bypass Sense**: When set to Active_Low, the BYPASS pin is active-Low. BYPASS has a parameter to control its active sense because an historical implementation made significant speed gains with an active-Low BYPASS, instead of active-High BYPASS. This is no longer necessarily the case, as sometimes active-High is as efficient, or more so. The details depend on the exact set of parameters.
- **Borrow In/Out Sense**: When set to Active_Low, the C_IN and C_OUT pins are active-Low for subtraction. This conforms to the legacy fabric implementation where this was an optimal setting. By setting **Borrow In/Out Sense** to Active_High, an active-High C_IN and C_OUT on subtraction is obtained.
- **Clock Enable**: When set to TRUE, the module is generated with a clock enable input.
- **Power on Reset Init Value**: Specifies (in hex) the value the S register initializes to during power-up reset.
- **Synchronous Clear**: Specifies if an SCLR pin is to be included.
- **Synchronous Set**: Specifies if an SSET pin is to be included. SSET pin is not valid in DSP slice implementations. See **Synchronous Set and Clear (Reset) Priority** for SCLR/SSET priorities.
- **Synchronous Init**: Specifies if an SINIT pin is to be included which, when asserted, synchronously sets the output value to the value defined by **Init Value**. Note that if SINIT is present, then neither SSET nor SCLR are present. SINIT pin is not valid in DSP slice implementations.
- **Init Value**: Specifies in hex the value the output initializes to when SINIT is asserted. Ignored if **Synchronous Init** = false.
- **Synchronous Controls and Clock Enable (CE) Priority**: This parameter controls whether or not the SCLR (and if fabric: SSET and SINIT) inputs are qualified by CE. When set to Sync_Overrides_CE, the synchronous controls override the CE signal. When set to CE_Overrides_Sync, the control signals have an effect only when CE is High. Note that on the fabric primitives, the SCLR and SSET controls override CE, so choosing CE_Overrides_Sync generally results in extra logic.
- **Sync Set and Clear (Reset) Priority**: Controls the relative priority of SCLR and SSET. When set to Reset_Overrides_Set, SCLR overrides SSET. The default is Reset_Overrides_Set, as this is the way the primitives are arranged. Making SSET take priority requires extra logic.

- **Latency Configuration**: Automatic sets optimal latency for maximum speed; Manual allows user to set Latency to one of the allowed values.
- **Latency**: Value used for latency when Latency Configuration is set to Manual. See the section, Pipelined Operation in Chapter 3 for more information.

## User Parameters

Table 4-1 shows the relationship between the fields in the Vivado IDE and the User Parameters (which can be viewed in the Tcl Console).

*Table 4-1:* **Vivado IDE Parameter to User Parameter Relationship**

| Vivado IDE Parameter | User Parameter | Default Value |
|---|---|---|
| Implement using | implementation | Fabric |
| A Input Type | a_type | Signed |
| B Input Type | b_type | Signed |
| A Input Width | a_width | 15 |
| B Input | b_width | 15 |
| Add Mode | add_mode | Add |
| Output Width | out_width | 16 |
| Latency Configuraton | latency_configuration | Manual |
| Latency | latency | 1 |
| Constant Input | b_constant | False |
| Constant Value | b_value | 000000000000000 |
| Clock Enable (CE) | ce | True |
| Carry In(C_IN) | c_in | False |
| Carry_Out(C_OUT) | c_out | False |
| Borrow In/Out Sense | borrow_sense | Active_Low |
| Synchronous Clear(SCLR) | sclr | False |
| Synchronous Set(SSET) | sset | False |
| Synchronous Init(SINIT) | sinit | False |
| Init Value (Hex) | sinit_value | 0 |
| Bypass | bypass | False |
| Bypass Sense | bypass_sense | Active_High |
| Synchronous Set and Clear(Reset) Priority | sync_ctrl_priority | Reset_Overrides_Set |
| Synchronous Controls and Clock Enable (CE) Priority | sync_ce_priority | Sync_Overrides_CE |

*Table 4-1:* ***(Cont'd)*Vivado IDE Parameter to User Parameter Relationship**

| Vivado IDE Parameter | User Parameter | Default Value |
|---|---|---|
| Bypass and Clock Enable (CE) Priority | bypass_ce_priority | CE_Overrides_Bypass |
| Power-on Reset Init Value (Hex) | ainit_value | 0 |

### Core Use through the Vivado Design Suite

The Vivado IP catalog performs error-checking on all input parameters. Resource estimation and latency information are also available.

Several files are produced when a core is generated, and customized instantiation templates for Verilog and VHDL design flows are provided in the `.veo` and `.vho` files, respectively. For detailed instructions, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 2].

### Core Use through System Generator for DSP

The Adder/Subtracter core is available through Xilinx System Generator for DSP, a design tool that enables the use of The MathWorks model-based design environment Simulink® software for FPGA design. The Adder/Subtracter core is one of the DSP building blocks provided in the Xilinx blockset for the Simulink software. The core can be found in the Xilinx Blockset in the Math section. The block is called AddSub. See the *System Generator for DSP User Guide* (UG640) [Ref 4] for more information.

### Output Generation

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 2].

## Constraining the Core

There are no constraints associated with this core.

## Simulation

For comprehensive information about Vivado® simulation components, as well as information about using supported third-party tools, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 5].

**IMPORTANT:** *For cores targeting 7 series or Zynq-7000 devices, UNIFAST libraries are not supported. Xilinx IP is tested and qualified with UNISIM libraries only.*

## Synthesis and Implementation

For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 2].

# Example Design

No example design is provided for this core.

# Test Bench

No demonstration test bench is provided for this core.

# Upgrading

This appendix contains information about migrating a design from ISE® to the Vivado® Design Suite, and for upgrading to a more recent version of the IP core. For customers upgrading in the Vivado Design Suite, important details (where applicable) about any port changes and other impact to user logic are included.

## Migrating to the Vivado Design Suite

### Updating from Adder/Subtracter v9.0 and Later

The Vivado Design Suite IP update feature can be used to update an existing Adder/Subtracter to version 12.0 of the core. The core can then be regenerated to create a new netlist. For information about migrating to the Vivado Design Suite, see the *ISE to Vivado Design Suite Migration Guide* (UG911) [Ref 6].

### Updating from Versions Prior to Adder/Subtracter v9.0

It is not currently possible to automatically update versions of the Adder/Subtracter core prior to v9.0. Some previous features and configurations are unavailable in Adder/Subtracter v12.0. Also, some port names differ between versions.

**RECOMMENDED:** *Use the Adder/Subtracter v12.0 Vivado IDE in the Vivado Design Suite to customize a new core.*

## Upgrading in the Vivado Design Suite

This section provides information about any changes to the user logic or port designations that take place when you upgrade to a more current version of this IP core in the Vivado Design Suite.

### Parameter Changes

There are no parameter changes in Adder/Subtracter v12.0 compared to v9.0 and later.

## Port Changes

There are no port changes in Adder/Subtracter v12.0 compared to v9.0 and later.

## Functionality Changes

There are no changes in functionality in Adder/Subtracter v12.0 compared to v9.0 and later.

## Simulation

Starting with Adder/Subtracter v12.0 (2013.3 version), behavioral simulation models have been replaced with IEEE P1735 Encrypted VHDL. The resulting model is bit and cycle accurate with the final netlist. For more information on simulation, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 5].

# Debugging

This appendix includes details about resources available on the Xilinx® Support website and debugging tools.

## Finding Help on Xilinx.com

To help in the design and debug process when using the Adder/Subtracter, the Xilinx Support web page contains key resources such as product documentation, release notes, answer records, information about known issues, and links for obtaining further product support.

### Documentation

This product guide is the main document associated with the Adder/Subtracter. This guide, along with documentation related to all products that aid in the design process, can be found on the Xilinx Support web page or by using the Xilinx Documentation Navigator.

Download the Xilinx Documentation Navigator from the Downloads page. For more information about this tool and the features available, open the online help after installation.

### Answer Records

Answer Records include information about commonly encountered problems, helpful information on how to resolve these problems, and any known issues with a Xilinx product. Answer Records are created and maintained daily ensuring that users have access to the most accurate information available.

Answer Records for this core can be located by using the Search Support box on the main Xilinx support web page. To maximize your search results, use proper keywords such as

- Product name
- Tool messages
- Summary of the issue encountered

A filter search is available after results are returned to further target the results.

**Master Answer Record for the Adder/Subtracter**

AR: 54491

## Technical Support

Xilinx provides technical support in the Xilinx Support web page for this LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support if you do any of the following:

- Implement the solution in devices that are not defined in the documentation.
- Customize the solution beyond that allowed in the product documentation.
- Change any section of the design labeled DO NOT MODIFY.

To contact Xilinx Technical Support, navigate to the Xilinx Support web page.

# Debug Tools

There are many tools available to address IP core design issues. It is important to know which tools are useful for debugging various situations.

## Vivado Design Suite Debug Feature

The debug feature inserts logic analyzer and virtual I/O cores directly into your design. The debug feature also allow you to set trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be analyzed. This feature in the Vivado IDE is used for logic debugging and validation of a design running in Xilinx devices.

The Vivado logic analyzer is used to interact with the logic debug LogiCORE IP cores, including:

- ILA 2.0 (and later versions)
- VIO 2.0 (and later versions)

See *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 7].

Appendix C

# Additional Resources and Legal Notices

## Xilinx Resources

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

## Documentation Navigator and Design Hubs

Xilinx® Documentation Navigator provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open the Xilinx Documentation Navigator (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 the Xilinx Documentation Navigator, click the **Design Hubs View** tab.
- On the Xilinx website, see the Design Hubs page.

***Note:*** For more information on Documentation Navigator, see the Documentation Navigator page on the Xilinx website.

# References

These documents provide supplemental material useful with this product guide:

1. *Vivado® Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
2. *Vivado Design Suite User Guide: Designing with IP* (UG896)
3. *Vivado Design Suite User Guide: Getting Started* (UG910)
4. *System Generator for DSP User Guide* (UG640)
5. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
6. *ISE® to Vivado Design Suite Migration Guide* (UG911)
7. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)

# Revision History

The following table shows the revision history for this document.

| Date | Version | Revision |
|---|---|---|
| 02/04/2021 | 12.0 | Added Versal ACAP support. |
| 11/18/2015 | 12.0 | Added support for UltraScale+ families. |
| 04/02/2014 | 12.0 | Added link to resource utilization information. |
| 12/18/2013 | 12.0 | • Added UltraScale™ architecture support information.<br>• Added Simulation, Synthesis, Example Design and Test Bench chapters.<br>• Updated Migrating appendix. |
| 10/02/2013 | 12.0 | Minor updates to IP Facts table and Migrating appendix. Document version number advanced to match the core version number. |
| 03/20/2013 | 1.0 | Initial release as a product guide. This document replaces the *LogiCORE IP Adder/Subtracter Data Sheet (DS214)*. |

# Please Read: Important Legal Notices