AXI HBICAP v1.0

LogiCORE IP Product Guide

Vivado Design Suite

PG349 (v1.0) October 30, 2019

XILINX®

Table of Contents

Introduction

The Xilinx® AXI High Bandwidth Internal Configuration Access Port (HBICAP) LogiCORE™ IP core for the AXI Interface enables an embedded microprocessor, such as the MicroBlaze™ processor, to read and write the FPGA configuration memory through the internal configuration access port (ICAPEn). This enables you to write software programs that modify the circuit structure and functionality during the operation of the circuit.

Features

- Supports ICAPEn both in internal and external modes
- Supports STARTUPEn both in internal and external modes
- AXI4-Lite interface for control path
- Memory mapped AXI4 slave interface for read and write data paths with maximum bursts of 256 beats per transaction
- Optional AXI4-Stream master interface for read datapath with unlimited burst per transaction
- Supports write and read transfers up to 2^{30} bytes in size
- Cut-through write path with no separate control on initiating write transfers
- Lock bit option to hold the control on ICAP when ICAP is in sharing mode with other blocks
- Indicates the read FIFO full condition through an output that can be used to gate the ICAP clock
- Interrupts for write and read paths based on FIFO conditions
- Soft reset, FIFO only reset options
- Status register to indicate the configuration Done and EoS
- Supports ICAP abort operation
- Supports independent read FIFO and write FIFO disable options

Note: The ICAPE2 primitive is applicable for 7 series devices. The ICAPE3 primitive is applicable for UltraScale™ and UltraScale+™ devices.

IP Facts

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family[1]	UltraScale+, UltraScale, Zynq®-7000 SoC, and 7 series FPGAs
Supported User Interfaces	AXI4-Lite, memory mapped AXI4, and AXI4-Stream
Resources	Resource Utilization
Provided with Core	
Design Files	VHDL
Example Design	Not provided
Test Bench	Not provided
Constraints File	Xilinx Design Constraints (XDC)
Simulation Model	Not provided
Supported Software Driver	Standalone
Tested Design Flows[3]	
Design Entry	Vivado® Design Suite
Simulation	For supported simulators, see the Xilinx Design Tools: Release Notes Guide
Synthesis	Vivado Synthesis
Support	
All Vivado IP Change Logs	Master Vivado IP Change Logs: 72775
Provided by Xilinx at the Xilinx Support website	

Notes:

1. For a complete list of supported devices, see the Vivado IP catalog.
2. For more information, see *7 Series FPGAs Overview Advanced Product Specification* (DS180).
3. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

Overview

The AXI HBICAP core provides the interface necessary to transfer data to and from the ICAPEn primitive.

Write Path

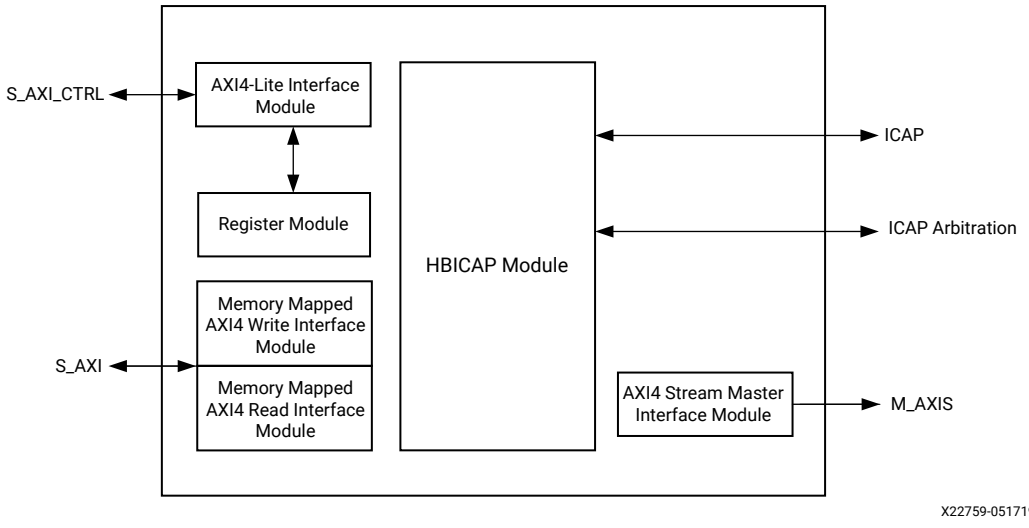
For writes to the ICAPEn, provide the necessary information, such as the transfer size into the AXI4-Lite based register. The data provided on the memory mapped AXI4 write channel is transferred into the ICAPEn interface passing through a write FIFO. Bursts up to 256 beats on the input memory mapped AXI4 write channel can be provided. Data is first stored in a write FIFO. The moment this FIFO is not empty, the data is transferred onto the ICAPEn interface.

Read Path

For reads from the ICAPEn, enable the read control bit and provide the transfer size. On the processor interface, select the memory mapped AXI4 slave interface or the AXI4-Stream master interface. With the AXI4-Stream interface, the IP initiates transaction and an unlimited burst of data can be read out. Both the write and read paths use the same size register to indicate the total number of words to be transferred. The status of the transfers is stored in the AXI4-Lite based registers and are explained in detail in subsequent sections.

The AXI HBICAP top-level block diagram is shown in the following figure.

Figure 1: **Top-level Block Diagram**



Module Descriptions

The AXI HBICAP core modules are described in this section.

AXI4-Lite Interface Module

The AXI4-Lite Interface Module provides the bidirectional interface between the HBICAP core and the AXI4 interface. The base element of the AXI4-Lite Interface Module is the slave attachment, which provides the basic functionality of AXI4 slave operation.

Memory Mapped AXI4 Read Interface Module

The memory mapped AXI4 Read Interface Module provides a read interface, which is used to read the data from the read FIFO in the read path. This is an optional interface that can be selected using the C_READ_MODE parameter. This is a full memory mapped AXI4 read slave interface with a fixed width of 32 bits and it accepts up to 16 outstanding read transactions.

Memory Mapped AXI4 Write Interface Module

The memory mapped AXI4 Write Interface Module provides a bidirectional write interface, which is used to write the data into the write FIFO in the write data path. This is a full memory mapped AXI4 write slave interface with a fixed width of 32 bits and accepts up to 16 outstanding write transactions.

AXI4-Stream Master Interface Module

The AXI4-Stream master interface module provides streaming reads. This is an optional interface and can be enabled using the C_READ_PATH parameter. This is a master interface and the IP initiates the packet once the read FIFO has some data and is not empty.

HBICAP Module

The HBICAP module provides the interface to the ICAPEn. It has a write FIFO, which stores the data locally. The data stored in the write FIFO is transferred to the ICAPE. The data that is read from the ICAPEn is stored in the read FIFO. The FIFO depth can be specified while customizing the IP.

ICAP Interface

The ICAP Interface is a point-to-point connection between the HBICAP controller and the ICAP primitive. The ICAP primitive enables read and write access to the registers inside the FPGA configuration system. The ICAP primitive and the behavior of the signals on this interface are described in the *UltraScale Architecture Configuration User Guide* (UG570)

This interface is exposed at the core level when the configuration primitives used by the IP (ICAP) are located outside the core (C_ICAP_EXTERNAL = 1).

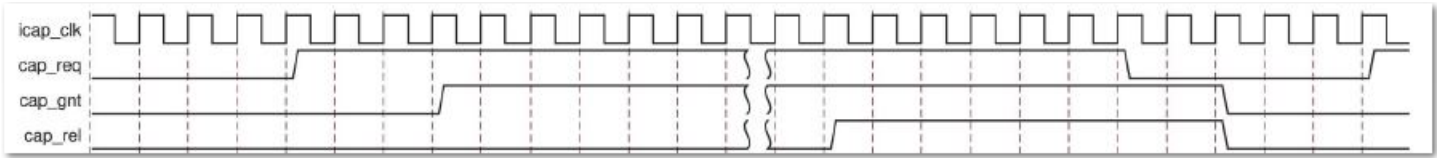
Note: The ICAPE2 primitive is applicable for 7 series devices. The ICAPE3 primitive is applicable for UltraScale™ and later devices.

ICAP Arbitration interface signals

The switching behavior is illustrated in the following figure. When the IP needs to access the ICAP primitive (in response to AXI4-Lite and memory mapped AXI4 transactions), the IP asserts the `cap_req` signal to request access to the ICAP primitive. It continuously asserts this signal as long as it needs access to the ICAP primitive. When the `cap_gnt` signal is High, the IP assumes that it has ICAP primitive control.

If the `cap_rel` signal is High, this indicates to the IP that it needs to release the ICAP primitive access. Subsequently, after finishing the ongoing transaction, the IP deasserts its `cap_req` signal. As shown in the figure, `cap_req` is asserted again (per a new AXI4-Lite and memory mapped AXI4 transaction request) only after three clock cycles have elapsed following `cap_rel` deassertion.

Figure 2: **Switching Behavior of ICAP Interface Signals**



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 about pricing and availability of other Xilinx® LogiCORE IP modules and tools, contact your local Xilinx sales representative.

Product Specification

Performance

Performance characterization is obtained using the margin system methodology. The details of the margin system characterization methodology are described in the Vivado IP Optimization (Fmax Characterization) appendix of the *Vivado Design Suite User Guide: Designing with IP* (UG896).

Part information: xcku040- ffva1156-2-e

Fixed clocks (MHz): icap_clk=125, s_axi_aclk=100, s_axi_mm_aclk=250, m_axis_aclk=250

Table 1: **Performance**

C_ICAP_EXTERNAL	C_ENABLE_ASYNC	Clock Input	Fmax (MHz)
0	1	m_axis_aclk	250
0	1	s_axi_mm_aclk	250
1	0	s_axi_aclk	125
0	0	s_axi_aclk	125

Resource Utilization

For full details about performance and resources use, see the following table:

Table 2: **Resource Utilization**

C_ICAP_EXTERNAL	C_ENABLE_ASYNC	Clock Input	LUTs	FFs	DSPs	36k BRAMs	18k BRAMs
0	1	m_axis_aclk	713	1580	0	1	1
0	1	s_axi_mm_aclk	713	1580	0	1	1
1	0	s_axi_aclk	819	1740	0	2	0
0	0	s_axi_aclk	768	1635	0	1	1

Port Descriptions

Input/Output Signals

The AXI4 HBICAP core I/O signals are listed in the following table.

Table 3: **I/O Signals**

Signal Name	Interface	I/O	Description
ICAP Interface Signals			
icap_clk[1]	ICAPEn	I	ICAPEn clock
eos_in[2]	NA	I	End of Start-up. This pin must be connected to a valid EOS signal if the Start-up primitive is not included in the HBICAP IP. This is ignored if the Start-up primitive is instantiated in the AXI HBICAP.
AXI4-Lite Bus Request and Qualifier Signals			
s_axi_aclk	AXI4-Lite	I	AXI4-Lite clock
s_axi_aresetn	AXI4-Lite	I	AXI reset; active-Low
s_axi_*	AXI4-Lite	I	See Appendix A of the *Vivado Design Suite: AXI Reference Guide* (UG1037) for AXI4 signals.
Memory Mapped AXI4 Bus Request and Qualifier Signals			
⚠ **CAUTION!** *When the S_AXI interface (memory mapped FIFO) is connected to the bus master through the interconnect where other master and slave interfaces have different data widths, data loss might occur because of an up-sized interconnect. Xilinx recommends using the LogiCORE IP AXI SmartConnect.*			
s_axi_mm_aclk	Memory mapped AXI4	I	Memory mapped AXI4 clock
s_axi_mm_resetn	Memory mapped AXI4	I	Memory mapped AXI4 reset, active-Low
s_axi_mm_*	Memory mapped AXI4	I/O	See Appendix A of the *Vivado Design Suite: AXI Reference Guide* (UG1037) for AXI4 signals.
AXI4-Stream Bus Signals			
m_axis_aclk	AXI4-Stream	I	AXI4-Stream clock
m_axis_aresetn	AXI4-Stream	I	AXI4-Stream reset, active-Low
m_axis_read_tvalid	AXI4-Stream	O	Indicates valid data is present on m_axis_read_tdata and m_axis_read_tlast ports. Default value is 0.
m_axis_read_tready	AXI4-Stream	I	Handshake signal for m_axis_read_tvalid
m_axis_read_tdata[31:0]	AXI4-Stream	O	Indicates the payload. Default value is 0.
m_axis_read_tlast	AXI4-Stream	O	Indicates the end of packet. Default value is 0.
System Signals			
ip2intc_irpt	AXI	O	Interrupt signal; default state is 0
ICAP Signals[3]			
icap_csib	ICAP	O	Active-Low ICAP enable

Table 3: **I/O Signals** *(cont'd)*

Signal Name	Interface	I/O	Description
icap_o	ICAP	O	ICAP data output bus
icap_i	ICAP	I	ICAP data input bus
icap_rdwrb	ICAP	O	ICAP read write select input. Active-High indicates read, active-low indicates write. The initial state is 0.
gate_icap_clk	ICAP	O	Indicates that the read FIFO is full and any further data received will be lost. This signal should be used at the system level to gate the ICAP clock to avoid data loss. Default value is 0.
icap_avail	ICAP	I	UltraScale™ and UltraScale+™ only. Indicates that ICAPEn is ready for accepting any transactions. This signal is applicable only when C_ICAP_EXTERNAL is 0. Default value is 1.
ICAP Arbiter Signals[3]			
cap_gnt	ICAP_ARBITER	I	For use with an ICAP arbiter. This signal is asserted by the arbiter to inform the HBICAP controller that it has permission to access the ICAP. After cap_gnt is asserted, it should remain asserted until cap_req is deasserted. If arbitration is not required, tie this signal to a constant 1.
cap_rel	ICAP_ARBITER	I	This signal indicates to the IP that it should relinquish control of the ICAP at the earliest safe opportunity. If the cap_rel signal is High, and if `cap_req` is already set by IP, then after finishing the ongoing transaction, the IP deasserts its `cap_req` signal.
cap_req	ICAP_ARBITER	O	For use with an ICAP arbiter. This signal is asserted by the IP on every clock cycle where it has data to transfer to the ICAP. The initial state is 0.

Notes:

1. 7 series devices have a limiting frequency of maximum 100 MHz on `icap_clk`. In synchronous mode, this port is unused inside the core (that is, in this mode, the core uses only `s_axi_aclk`). For more details, see the DC and switching characteristics documents *Artix-7 FPGAs Data Sheet: DC and AC Switching Characteristics* (DS181), *Kintex-7 FPGAs Data Sheet: DC and AC Switching Characteristics* (DS182), and *Virtex-7 FPGAs Data Sheet: DC and AC Switching Characteristics* (DS183).

   For UltraScale and UltraScale+ devices, the maximum frequency on `icap_clk` varies across devices. For more details, see the DC and switching characteristics documents *Kintex UltraScale FPGAs Data Sheet: DC and AC Switching Characteristics* (DS892), *Kintex UltraScale+ FPGAs Data Sheet: DC and AC Switching Characteristics* (DS922), and *Virtex UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS923).
2. This input port is used only in 7 series devices when STARTUPE2 is not included in the IP. In other configurations, this port is unused inside the core.
3. These signals are valid only if the external ICAP is used (C_ICAP_EXTERNAL = 1).

Register Space

The register address mapping of the AXI4 HBICAP is offset from the base address C_S_AXI_CTRL_BASEADDR . The AXI HBICAP internal register set is described in the following table.

Note: The AXI4-Lite write access register is updated by the 32-bit AXI Write Data (`*_wdata`) signal, and is not impacted by the AXI Write Data Strobe (`*_wstrb`) signal. For a write, both the AXI Write Address Valid (`*_awvalid`) and AXI Write Data Valid (`*_wvalid`) signals should be asserted together.

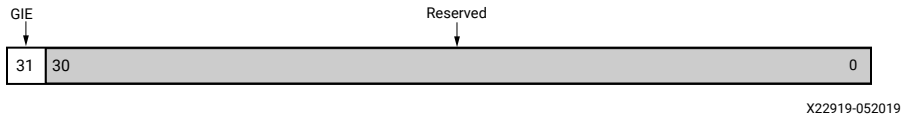
Table 4: **Register Address Space**

Address (Hex)	Register
1Ch	Global Interrupt Enable Register
20h	Interrupt Status Register
028h	IP Interrupt Enable Register
108h	Size Register
10Ch	Control Register
110h	Status Register
114h	Write FIFO Vacancy Register
118h	Read FIFO Occupancy Register
11Ch	Abort Status Register

Global Interrupt Enable Register

The Global Interrupt Enable Register (GIER) register shown in the following figure is used to globally enable the final interrupt output from the interrupt controller. This bit is a read/write bit and is cleared upon reset.

Figure 3: **Global Interrupt Enable Register (GIER)**



The bit definitions for the register are shown in the following table.

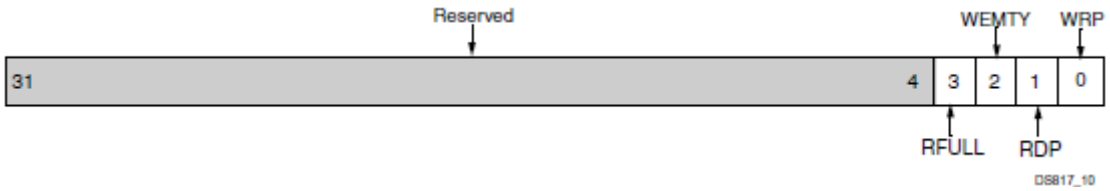
Table 5: **Global Interrupt Enable Register Bit Definitions (01Ch)**

Bit	Field Name	Access Type	Description
31	GIE	Read/Write	0 = Disabled 1 = Enabled Reset value=0
30–0	Reserved	N/A	Reserved

IP Interrupt Status Register

Four unique interrupt conditions are possible in the HBICAP core. The interrupt controller has a register that can enable each interrupt independently. The bit assignment in the Interrupt register for a 32-bit data bus is shown in the following figure.

Figure 4: **IP Interrupt Status Register (IPISR)**



The interrupt register is a read/toggle on write register. By writing a 1, it toggles the value of the bit. All register bits are cleared upon reset. The bit definitions for the register are shown in the following table.

Table 6: **IP Interrupt Status Register Bit Definitions**

Bit	Field Name	Access Type	Description
31–4	Reserved	N/A	Reserved
3	RFULL	R/TOW[1]	Read FIFO full Reset value=0
2	WEMPTY[2]	R/TOW[1]	Write FIFO empty Reset value=0
1	RDP	R/TO[1]W	Interrupt set and remains set if the read FIFO occupancy is greater than half of the read FIFO depth Reset value=0
0	WRP[2]	R/TOW[1]	Interrupt set and remains set if the write FIFO occupancy is less than half of the write FIFO depth Reset value=0

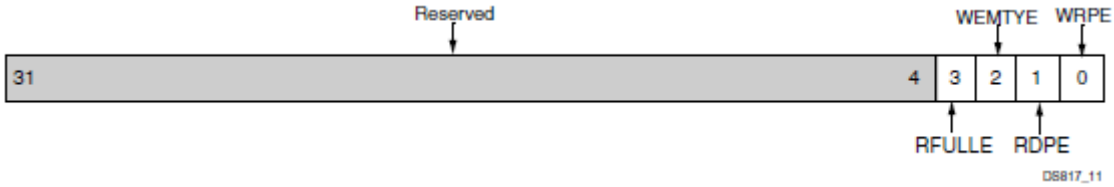
Notes:

1. TOW = toggle on write. Writing a 1 to a bit position within the register causes the corresponding bit position in the register to toggle.
2. WRP and WEMTY bits are not important when the write FIFO is disabled. Xilinx recommends that these interrupts be disabled when HBICAP is configured in Lite Mode.

IP Interrupt Enable Register

The IP Interrupt Enable Register (IPIER) has an enable bit for each defined bit of the IPISR as shown in the following figure. All bits are cleared upon reset.

Figure 5: **IP Interrupt Enable Register (IPIER)**



The bit definitions for the register are shown in the following table.

Table 7: **IP Interrupt Enable Register Bit Definitions (028h)**

Bit	Field Name	Access Type	Description
31–4	Reserved	N/A	Reserved
3	RFULLE	Read/ Write[1]	Read FIFO full interrupt enable Reset value=0
2	WEMTYE	Read/ Write[1]	Write FIFO empty interrupt enable Reset value=0
1	RDPE	Read/ Write[1]	Read FIFO occupancy greater than half of its depth interrupt enable Reset value=0
0	WRPE	Read/ Write[1]	Write FIFO occupancy less than half of its depth interrupt enable Reset value=0

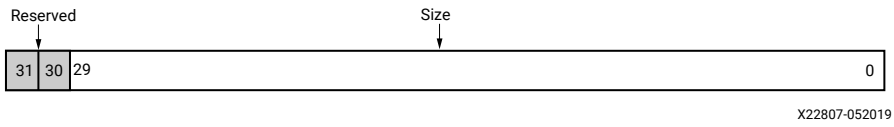
Notes:

1. Writing a 1 to this bit enables the particular interrupt. Writing a 0 to this bit disables the particular interrupt.

Size Register

The Size (SZ) register shown in the following figure is a 30-bit write-only register that determines the number of 32-bit words to be transferred from the ICAPEn to the read FIFO and from the write FIFO to the ICAP. (This signifies the number of 32-bit data beats that are expected.)

Figure 6: **Size Register**



The bit definitions for the register are shown in the following table.

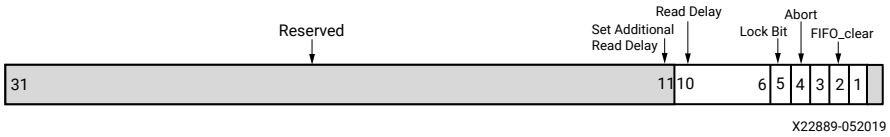
Table 8: **Size Register Bit Definitions (108h)**

Bit	Field Name	Access Type	Description
31–30	Reserved	N/A	Reserved
29-0	Size	Read/Write	Number of words to be transferred from the ICAPEn to the FIFO

Control Register

The Control Register (CR) shown in the following figure is a 32-bit read/write register that determines the direction of the data transfer. It controls whether a configuration or a readback occurs. Writing to this register initiates the transfer.

Figure 7: **Control Register**



The bit definitions for the register are shown in the following table.

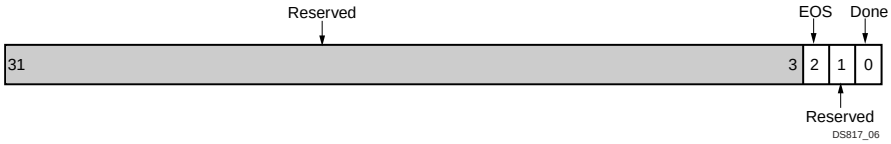
Table 9: **Control Register Bit Definitions (10Ch)**

Bit	Default Value	Access Type	Description
31–12	0	N/A	Reserved
11	0	Clear on Write	**Set Additional Read Delay** Set this bit to 1 to load value in next five bits
10-6	0	Read/Write	**Additional Read Delay** This field applies only to ICAP reads. This value indicates the number of additional clocks to be ignored by HBICAP between the RDWRB signal assertion until the ICAP puts the valid data. The default is 0.
5	0	Read/Write	**Lock bit** 0 = Unlock, cap_req does not depend on this bit 1 = Lock, cap_req output is ORed with this bit, which locks the access to the ICAP
4	0	Read/Write	1 = Aborts the read or write of the ICAPEn and clears the FIFOs
3	0	Read/Write	**SW_reset** 1 = Resets all the registers
2	0	Read/Write	**FIFO_clear** 1 = Clears the FIFOs
1	0	Read/Write	**Read** 1 = Initiate ICAPEn Read
0	0	N/A	**Reserved**

Status Register

The Status Register (SR) shown in the following figure is a 32-bit read register that contains the ICAPEn status bits.

Figure 8: **Status Register (SR)**



The bit definitions for the register are shown in the following table.

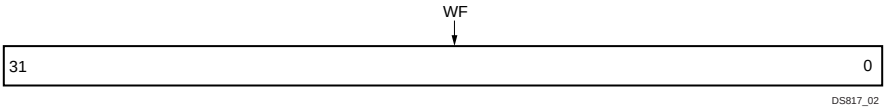
Table 10: **Status Register Bit Definitions (110h)**

Bit	Default Value	Access Type	Description
31–3	0	N/A	Reserved
2	3	Read	**EOS Bit**: Indicates that the EOS is complete. The ICAPEn can be accessed only when this bit is 1.
1	0	N/A	Reserved
0	0	Read	1 = Idle / Done with previous operation (configuration or read) 0 = Busy

Write FIFO Vacancy Register

The Write FIFO Vacancy (WFV) register shown in the following figure is a 32-bit read only register that indicates the vacancy of the write FIFO. The actual depth of the write FIFO is one less than the value specified during customization. For example, if the write FIFO depth is set to 1024 during customization, the actual FIFO depth is 1023 (or `0x3FF`). This register reports the actual write FIFO vacancy.

Figure 9: **Write FIFO Vacancy Register (WFV)**



The bit definitions for the register are shown in the following table.

Table 11: **Write FIFO Vacancy Register Bit Definitions (114h)**

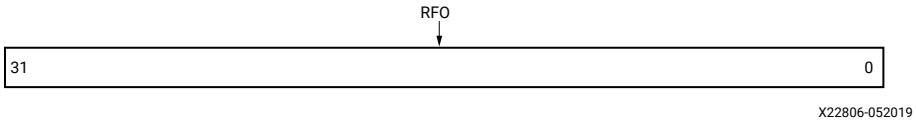
Bit	Default Value	Access Type	Description
31-0	Based on actual write FIFO	Read	**WFV**:Write FIFO Vacancy. This value is based on the actual write FIFO size.

***Note*:** This register is not important when the write FIFO is disabled.

Read FIFO Occupancy Register

The Read FIFO Occupancy (RFO) register shown in the following figure is a 32-bit read-only register that indicates occupancy of the read FIFO. The actual depth of the read FIFO is one less than the value specified during customization. For example, if the read FIFO depth is set to 256 during customization, the actual FIFO depth is 255 (or `0xFF`). This register reports the actual read FIFO occupancy. This register is not important when the disable read FIFO is checked when the read FIFO is disabled.

Figure 10: **Read FIFO Occupancy Register (RFO)**



The bit definitions for the register are shown in the following table.

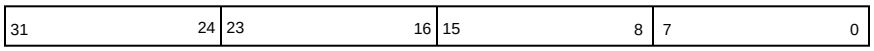
Table 12: **Read FIFO Occupancy Register Bit Definitions (118h)**

Bit	Default Value	Access Type	Description
31-0	0	Read	**RFO**:Read FIFO Occupancy. This value is based on the actual Write FIFO size.

Abort Status Register

An abort is an interruption that occurs in the configuration or read-back sequence when the state of `icap_rdwrb` changes while `icap_csib` is asserted. During a configuration abort, internal status is driven onto the `icap_i[7:0]` pins over the next four clock cycles. These four bytes are stored in the abort status register with the first status word available in the 31–24 bits while the last status word is in the 7–0 bits. After the abort sequence finishes, you can re-synchronize the configuration logic and resume configuration. Abort indicates the status of the ICAPEn during the configuration or reading the configuration. The Abort Status register is shown in the following figure.

Figure 11: **Abort Status Register**



The bit definitions for the register are shown in the following table.

Table 13: **Abort Status Register (11Ch)**

Bit	Default Value	Access Type	Description
31–24	0	Read	**Status 0**: First Abort Status word
23-16	0	Read	**Status 1**: Second Abort Status word
15-8	0	Read	**Status 2**: Third Abort Status word
7-0	0	Read	**Status 3**: Fourth Abort Status Word

Designing with the Core

This section includes guidelines and additional information to facilitate designing with the core.

General Design Guidelines

Registering Signals

To simplify timing and increase system performance in a programmable device design, keep all inputs and outputs registered between the user application and the core. This means that all inputs and outputs from the user application should come from, or connect to, a flip-flop. While registering signals might not be possible for all paths, it simplifies timing analysis and makes it easier for the Xilinx® tools to place and route the design.

Recognize Timing Critical Signals

The constraints provided with the example design identify the critical signals and timing constraints that should be applied.

Make Only Allowed Modifications

You should not modify the core. Any modifications can have adverse effects on system timing and protocol compliance. Supported user configurations of the core can only be made by selecting the options in the customization IP dialog box when the core is generated.

Clocking

The core works in three or four clock domains depending on whether the AXI4-Stream master interface is enabled: `s_axi_aclk`, `s_axi_mm_aclk`, `icap_clk`, and, optionally, `m_axis_aclk`.

Resets

The active-Low reset, `s_axi_aresetn`, is synchronized to `s_axi_aclk clock`. The number of resets are either two or three depending on whether the AXI4-Stream master interface is enabled or not. The active-Low reset, `s_axi_mm_aresetn`, is synchronized to `s_axi_mm_aclk`. The active-Low reset, `m_axis_resetn`, is synchronized to `m_axis_aclk`.

Programming Sequence

Every read ICAPEn operation is a write-read-write sequence. The first write contains SYNC and other commands that initiate the read operation on the ICAPEn, followed by the actual read operation, followed by a second write that contains DE-SYNC commands. The following steps describe this programming sequence.

Read Programming Sequence

1. Program the Size register with the number of words you want to write.
2. Send the first set of words you want to write to the `ICAPEn` using the memory mapped AXI4 interface using burst transactions.
3. Wait for the Done signal from the Status register, which indicates that the requested number of words have been written on the `ICAPEn` interface.
4. Program the Size register again with the number of words to be read from the `ICAPEn`.
5. Program the Control register with a value of `0x00000002`, which initiates a read on the `ICAPEn`.
6. Use the read interfaces in one of the following ways.
   - **Read using memory mapped AXI4 read burst transactions:** Initiate memory mapped AXI4 read burst transactions and continue the process until the required number of bytes are read out.
   - **Read using the AXI4-Stream interface:** In this mode, the HBICAP core initiates the stream transactions. Wait until TLAST, which indicates the end of the transfer.
7. Hardware clears the Control register bits after the successful completion of the data transfer from the ICAPEn to the read FIFO.
8. Software should not initiate another read or configuration to the `ICAPEn` until the read bit in the Control register is cleared.
9. Program the Size register with a second set of writes, which contains DE-SYNC and other commands to terminate the Read operation on the `ICAPEn`.

10. Send the second set of words to be written to the `ICAPEn` using the memory mapped AXI4 interface using burst transactions.
11. The Done signal from the Status register indicates that the requested number of words have been written on the `ICAPEn` interface.

Write Programming Sequence

1. Program the number of words to be transferred to the Size register.
2. Send burst transactions from the memory mapped AXI4 interface. A maximum burst of 256 beats or words can be sent per transaction.
3. Monitor the Done bit in the Status register. This bit is set to '1' after all words mentioned in the Size register are put on the ICAPEn interface and continue the process until all bitstream words are written to the ICAPEn.

For more information, see Status Register and Control Register.

Abort

1. Initiate a write or read of the ICAPEn using the steps in Programming Sequence, and while waiting for the completion of the operation, perform the following steps.
2. Write a value of `0x00000010` to the Control register to initiate an abort.
3. The Done bit in the Status register indicates whether the abort operation is completed.
4. Read the Abort Status register that contains the four bytes read from the ICAPEn, which indicates the status of the abort operation.
5. The hardware clears the Control register bits after the successful completion of the abort-on read, abort-on configuration, or normal abort.
6. The software should not initiate another read or configuration to the ICAPEn until the abort bit in the Control register is cleared.

IMPORTANT! *The core uses the ICAPE2 primitive in 7 series devices or the ICAPE3 primitive in UltraScale and UltraScale+ devices. The ICAPEn port interface is similar to the SelectMAP interface but is accessible from general interconnects rather than the device pins. The JTAG or boundary scan configuration mode pin setting (M2:M0 = `101`) disables the ICAPEn interface. If JTAG is used as the primary configuration method, another mode pin setting must be selected to avoid disabling the ICAPEn. JTAG configuration remains available because it overrides other means of configuration and the core functions as intended. The ICAPEn is also disabled if the persist bit in the device configuration logic control register is set. There can be cases when the ICAPEn is accessed as soon as JTAG programming is complete. In such cases, the ICAPEn is not accessible because JTAG might be performing other tasks.*

RECOMMENDED: *Allow sufficient time before initiating any transaction to ICAPEn, or be sure to enable the use of the STARTUPE2 or STARTUPE3 primitive within the IP.*

Design Flow Steps

This section 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)
- *Vivado Design Suite User Guide: Designing with IP* (UG896)
- *Vivado Design Suite User Guide: Getting Started* (UG910)
- *Vivado Design Suite User Guide: Logic Simulation* (UG900)

Customizing and Generating the Core

This section includes information about using Xilinx® tools to customize and generate the core in the Vivado® Design Suite.

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) 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, 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:

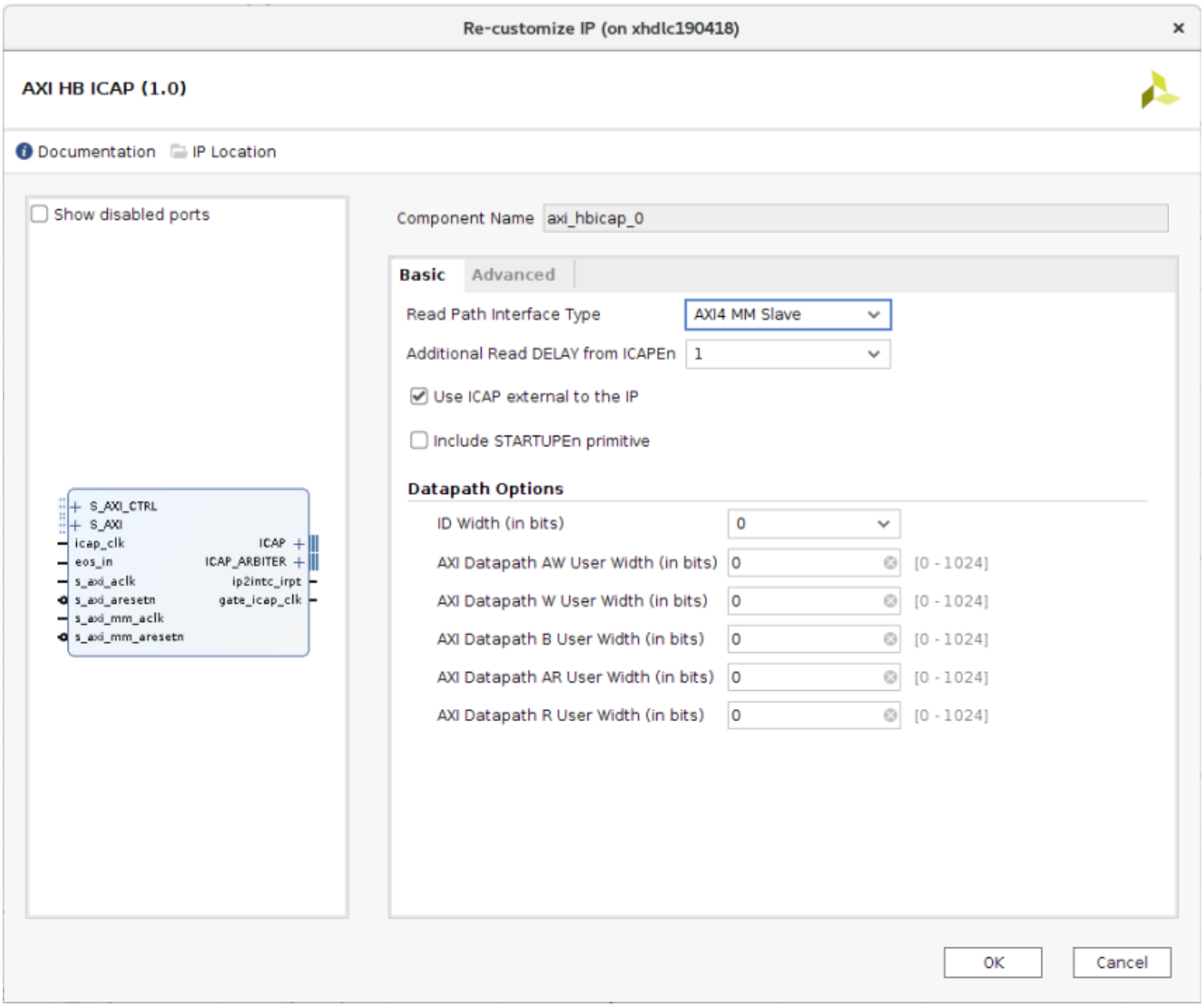
1. Select the IP from the IP catalog.
2. Double-click the selected IP or select the Customize IP command from the toolbar or right-click menu.

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

Figures in this chapter are illustrations of the Vivado IDE. The layout depicted here might vary from the current version.

Basic Tab

Figure 12: **Basic Tab**

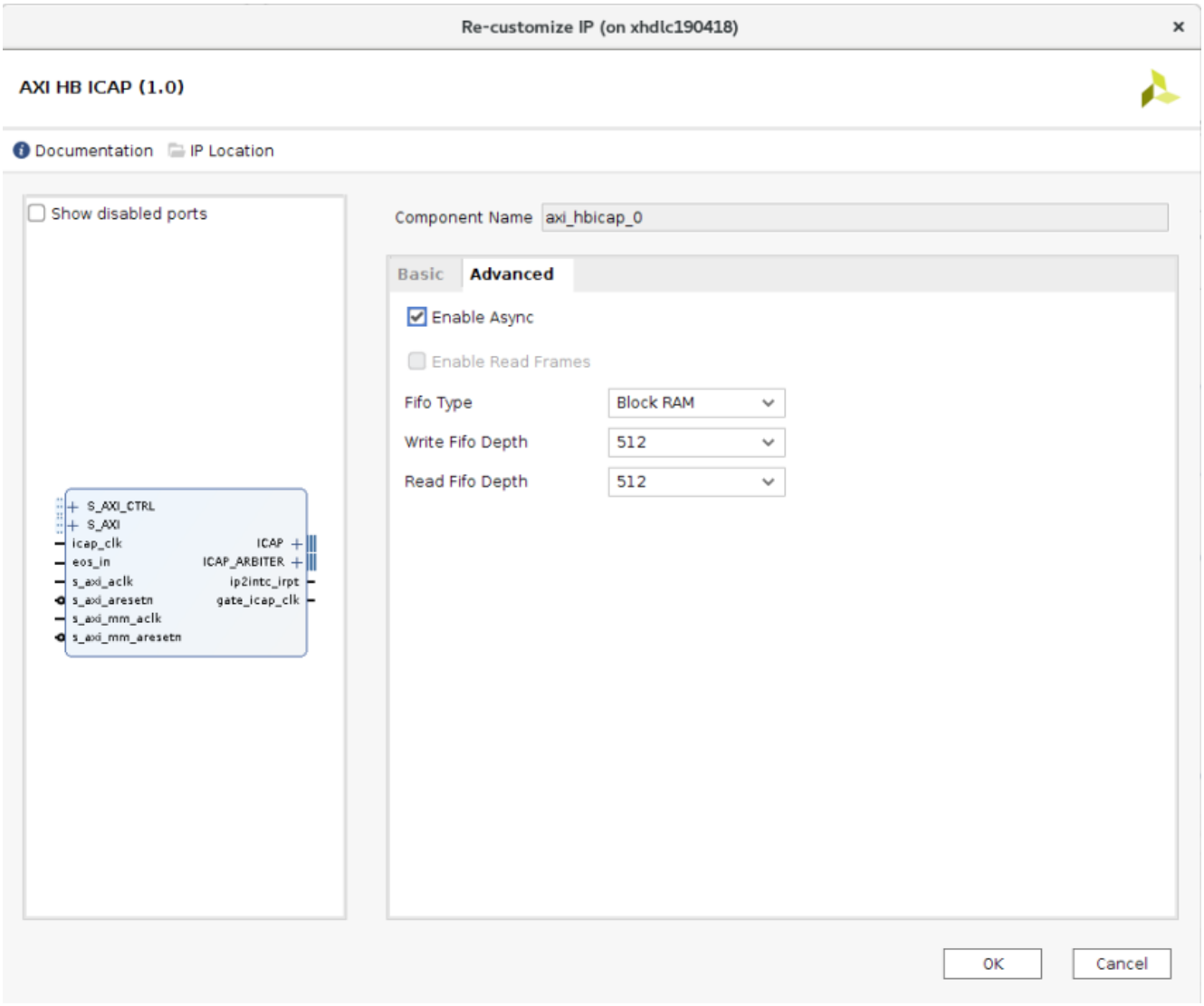


- **Read Path Interface Type:** Select read path interface from the AXI4 MM Slave and AXI4 Stream options.
- **Additional Read DELAY from ICAPEn:** Add required additional delay based on number of stages between IP and ICAPEn primitive when connected externally.
- **Instantiate STARTUPEn Primitive:** Select this option to instantiate the STARTUPEn primitive. The EOS output from STARTUPEn is used to control access to ICAPEn. EOS output signifies the end of device configuration. ICAPEn can be accessed only after EOS goes High.
- **Use ICAP External to the IP:** Select this option to connect the IP to an external ICAPEn primitive, shared with other IPs.

Advanced Tab

The advanced options that affect the AXI HBICAP core are shown in the following figure.

Figure 13: **Advanced Tab**



- **Enable Aysnc:**

  Select this box if the `icap_clk` and AXI clocks are asynchronous. In synchronous mode, only `s_axi_aclk` is used.

- **Enable Read Frames:** Select this box only when performing large frame reads from the ICAPEn. This instantiates an extra BUFGCTRL within the IP.

- **FIFO Type:** Select between the block RAM or distributed RAM type of FIFO to instantiate. Select this box only when performing large frame reads from the ICAPEn. This instantiates an extra BUFGCTRL within the IP when ICAPEn is in internal mode.
- **Write FIFO Depth:** Select the depth of the write FIFO from the available values of 64, 128, 256, 512, 1024, 2048, and 4096. The actual depth of the write FIFO is one less than the one specified during core customization.
- **Read FIFO Depth:** Select the depth of the read FIFO from the available values of 64, 128, 256, 512, 1024, 2048, and 4096. The actual depth of the read FIFO is one less the value specified during core customization.

User Parameters

The following table shows the relationship between the fields in the Vivado® IDE and the user parameters (which can be viewed in the Tcl Console).

Table 14: **User Parameters**

Vivado IDE Parameter/Value	User Parameter/Value	Default Value
Use ICAP external to the IP	C_ICAP_EXTERNAL	0
Include STARTUPEn primitive	C_INCLUDE_STARTUP	0
Enable Async	C_ENABLE_ASYNC	1
Enable read frames	C_OPERATION	0
FIFO type	C_BRAM_SRL_FIFO_TYPE	1
Write FIFO depth	C_WRITE_FIFO_DEPTH	64
Read FIFO depth	C_READ_FIFO_DEPTH	64
Share the unused STARTUP ports	C_SHARED_STARTUP	0
Read path Interface type	C_READ_PATH	0 (AXI4 MM Slave)
Additional read delay from ICAPEn	C_READ_DELAY	1

Output Generation

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

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).

Synthesis and Implementation

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

Appendix A

Debugging

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

If the IP requires a license key, the key must be verified. The Vivado® design tools have several license checkpoints for gating licensed IP through the flow. If the license check succeeds, the IP can continue generation. Otherwise, generation halts with an error. License checkpoints are enforced by the following tools:

- Vivado Synthesis
- Vivado Implementation
- write_bitstream (Tcl command)

IMPORTANT! *IP license level is ignored at checkpoints. The test confirms a valid license exists. It does not check IP license level.*

Finding Help on Xilinx.com

To help in the design and debug process when using the core, 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. The Xilinx Community Forums are also available where members can learn, participate, share, and ask questions about Xilinx solutions.

Documentation

This product guide is the main document associated with the core. 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 keywords such as:

- Product name
- Tool message(s)
- Summary of the issue encountered

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

Technical Support

Xilinx provides technical support on the Xilinx Community Forums 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 ask questions, navigate to the Xilinx Community Forums.

Debug Tools

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

Vivado Design Suite Debug Feature

The Vivado® Design Suite debug feature inserts logic analyzer and virtual I/O cores directly into your design. The debug feature also allows 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 the *Vivado Design Suite User Guide: Programming and Debugging* (UG908).

XILINX

Appendix B

Additional Resources and Legal Notices

Xilinx Resources

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

Documentation Navigator and Design Hubs

Xilinx® Documentation Navigator (DocNav) provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open DocNav:

- From the Vivado® IDE, select **Help → Documentation and Tutorials**.
- On Windows, select **Start → All Programs → Xilinx Design Tools → DocNav**.
- At the Linux command prompt, enter `docnav`.

Xilinx Design Hubs provide links to documentation organized by design tasks and other topics, which you can use to learn key concepts and address frequently asked questions. To access the Design Hubs:

- In DocNav, click the **Design Hubs View** tab.
- On the Xilinx website, see the Design Hubs page.

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

References

These documents provide supplemental material useful with this guide:

1. *7 Series FPGAs Data Sheet: Overview* (DS180)
2. *Vivado Design Suite User Guide: Designing with IP* (UG896)
3. *Vivado Design Suite: AXI Reference Guide* (UG1037)
4. *Virtex-7 FPGAs Data Sheet: DC and AC Switching Characteristics* (DS183)
5. *Kintex-7 FPGAs Data Sheet: DC and AC Switching Characteristics* (DS182)
6. *Artix-7 FPGAs Data Sheet: DC and AC Switching Characteristics* (DS181)
7. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
8. *Vivado Design Suite User Guide: Designing with IP* (UG896)
9. *Vivado Design Suite User Guide: Getting Started* (UG910)
10. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
11. *ISE to Vivado Design Suite Migration Guide* (UG911)
12. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
13. *Vivado Design Suite User Guide: Implementation* (UG904)
14. *AXI Interconnect LogiCORE IP Product Guide* (PG059)
15. *MicroBlaze Debug Module (MDM) LogiCORE IP Product Guide* (PG115)
16. *AXI Traffic Generator LogiCORE IP Product Guide* (PG125)
17. *AMBA AXI and ACE Protocol Specification* (UG904)
18. *UltraScale Architecture Configuration User Guide* (UG570)

Revision History

The following table shows the revision history for this document.

Section	Revision Summary
10/30/2019 Version 1.0	
Initial release	N/A

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