AXI Bridge for PCI Express Gen3 Subsystem v2.1

Product Guide

Vivado Design Suite

PG194 June 8, 2016

XILINX
ALL PROGRAMMABLE™

Table of Contents

Chapter 5: Example Design

Chapter 6: Test Bench

Appendix A: Debugging

Appendix B: Migrating and Upgrading

Appendix C: Additional Resources and Legal Notices

Introduction

The Xilinx® AXI Bridge for PCI Express® Gen3 core is an interface between AXI4 and PCI Express.

Features

- Supports UltraScale™ architecture and Virtex®-7 XT FPGA Gen3 Integrated Blocks for PCI Express
- Maximum Payload Size (MPS) up to 512 bytes
- Multiple Vector Messaged Signaled Interrupts (MSIs)
- MSI-X interrupt support
- Legacy interrupt support
- Memory-mapped AXI4 access to PCIe® space
- PCIe access to memory-mapped AXI4 space
- Tracks and manages Transaction Layer Packets (TLPs) completion processing
- Detects and indicates error conditions with interrupts
- Optimal AXI4 pipeline support for enhanced performance
- Compliant with Advanced RISC Machine (ARM®) Advanced Microcontroller Bus Architecture 4 (AMBA®) AXI4 specification
- Supports up to six PCIe 32-bit or three 64-bit PCIe Base Address Registers (BARs) as Endpoint
- Supports up to two PCIe 32-bit or a single PCIe 64-bit BAR as Root Port

Facts Table	
Core Specifics	
Supported Device Family	UltraScale Architecture, Virtex-7 XT(2)
Supported User Interfaces	AXI4
Resources	Performance and Resource Utilization web page
Provided with Core	
Design Files	Verilog
Example Design	Verilog
Test Bench	Verilog
Constraints File	Xilinx Design Constraints (XDC)
Simulation Model	Not Provided
Supported S/W Driver	Not Provided
Tested Design Flows(1)	
Design Entry	Vivado® Design Suite
Simulation	For supported simulators, see the Xilinx Design Tools: Release Notes Guide
Synthesis	Vivado synthesis
Support	
Provided by Xilinx at the Xilinx Support web page	

Notes:

1. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.
2. Except for XC7VX485T, XC7V585T, and XC7V2000T, Virtex-7 devices are supported.

Overview

The AXI Bridge for PCI Express Gen3 core is designed for the Vivado® IP integrator in the Vivado Design Suite. The AXI Bridge for PCI Express Gen3 core provides an interface between an AXI4 customer user interface and PCI Express using the Xilinx® Integrated Block for PCI Express. The AXI Bridge for PCI Express Gen3 core provides the translation level between the AXI4 embedded system to the PCI Express system. The AXI Bridge for PCI Express Gen3 core translates the AXI4 memory read or writes to PCIe® Transaction Layer Packets (TLP) packets and translates PCIe memory read and write request TLP packets to AXI4 interface commands.

The architecture of the AXI Bridge for PCI Express Gen3 is shown in Figure 1-1.

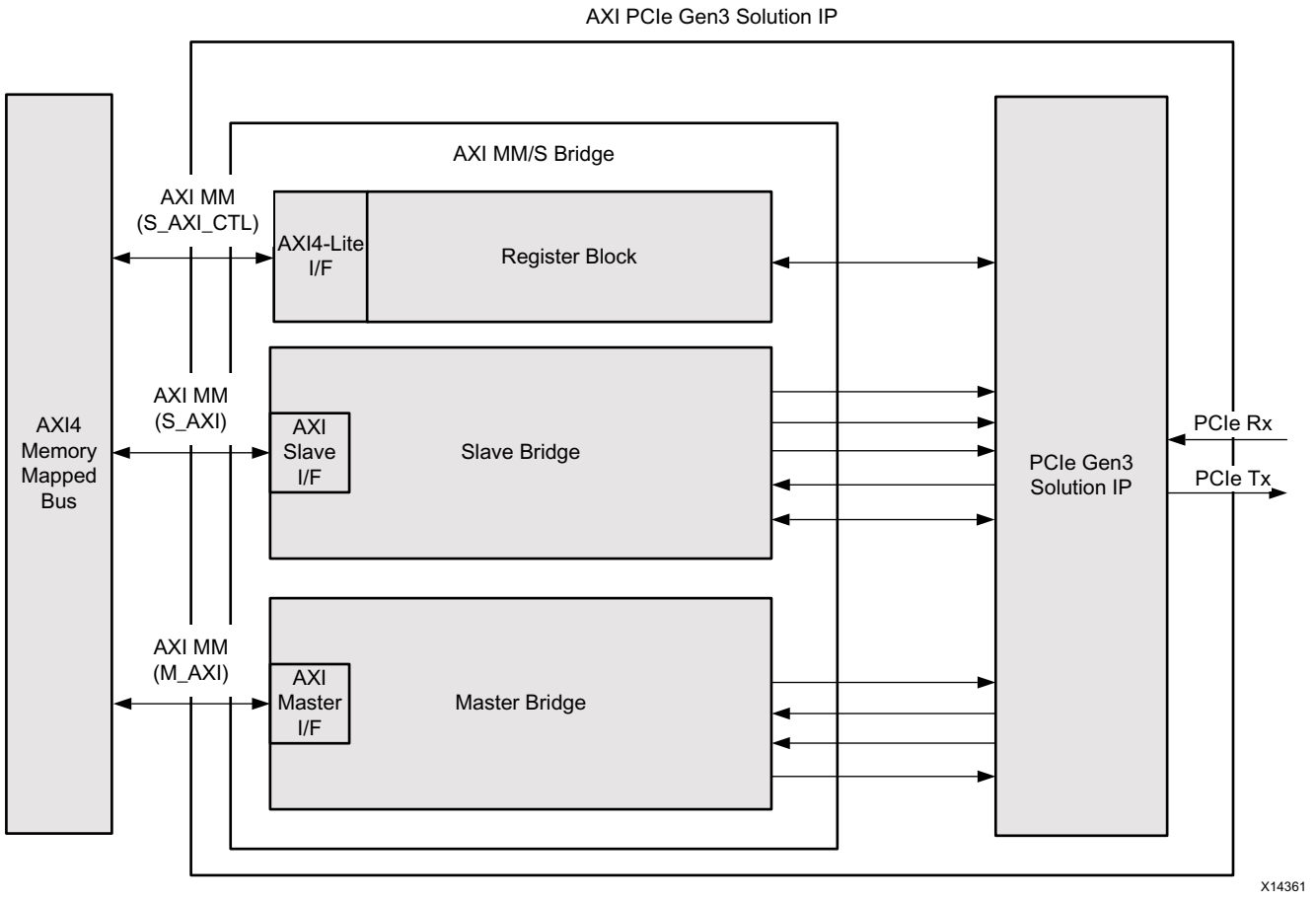

Figure 1-1: **High-Level AXI Bridge for PCI Express Gen3 Architecture**

Feature Summary

The AXI Bridge for PCI Express Gen3 core is an interface between the AXI4 and PCI Express. It contains the memory mapped AXI4 to AXI4-Stream Bridge and the AXI4-Stream Enhanced Interface Block for PCIe. The memory-mapped AXI4 to AXI4-Stream Bridge contains a register block and two functional half bridges, referred to as the Slave Bridge and Master Bridge. The slave bridge connects to the AXI4 Interconnect as a slave device to handle any issued AXI4 master read or write requests. The master bridge connects to the AXI4 Interconnect as a master to process the PCIe generated read or write TLPs. The core uses a set of interrupts to detect and flag error conditions.

The AXI Bridge for PCI Express Gen3 core supports both Root Port and Endpoint configurations.

- When configured as an Endpoint, the AXI Bridge for PCI Express Gen3 core supports up to six 32-bit or three 64-bit PCIe Base Address Registers (BARs).
- When configured as a Root Port, the core supports up to two 32-bit or a single 64-bit PCIe BAR.

The AXI Bridge for PCI Express Gen3 core is compliant with the PCI Express Base Specification v2.0 [Ref 10] and with the AMBA® AXI Protocol Specification [Ref 9].

Unsupported Features

The following features are not supported in the AXI Bridge for PCI Express Gen3 core.

- Tandem Configuration solutions (Tandem PROM, Tandem PCIe, Tandem with Field Updates, PR over PCIe) are not supported for Virtex-7 XT devices.
- Narrow burst

Limitations

PCIe Transaction Type

The PCIe transactions generated are those that are compatible with the AXI4 specification. Table 1-1 lists the supported PCIe transaction types.

Table 1-1: **Supported PCIe Transaction Types**

TX	RX
MRd32	MRd32
MRd64	MRd64
MWr32	MWr32
MWr64	MWr64
Msg(INT/Error)	Msg(SSPL,INT,Error)
Cpl	Cpl
CplD	CplD
Cfg Type0/1 (RP)	

PCIe Capability

The PCIe capabilities of the AXI Bridge for PCI Express Gen3 support only the following capabilities due to the AXI4 specification:

- 1 PF
- MSI
- MSI-X
- PM
- Advanced error reporting (AER)

Others

AXI Slave

- Only supports the INCR burst type. Other types result in the Slave Illegal Burst (SIB) interrupt.
- No memory type support (AxCACHE)
- No protection type support (AxPROT)
- No lock type support (AxLOCK)

- No non-contiguous byte enable support (WSTRB)

AXI Master

- Only issues the INCR burst type
- Only issues the Data, Non-secure, and Unprivileged protection type

Licensing and Ordering Information

This Xilinx module is provided at no additional cost with the Xilinx Vivado Design Suite under the terms of the Xilinx End User License.

Information about this and other Xilinx modules is available at the Xilinx Intellectual Property page. For information on pricing and availability of other Xilinx modules and tools, contact your local Xilinx sales representative.

For more information, visit the AXI Bridge for PCI Express Gen3 product page.

Product Specification

The Register block contains registers used in the AXI Bridge for PCI Express Gen3 core for dynamically mapping the AXI4 memory mapped (MM) address range provided using the AXIBAR parameters to an address for PCIe® range.

The slave bridge provides termination of memory-mapped AXI4 transactions from an AXI master device (such as a processor). The slave bridge provides a way to translate addresses that are mapped within the AXI4 memory mapped address domain to the domain addresses for PCIe. Write transactions to the Slave Bridge are converted into one or more MemWr TLPs, depending on the configured Max Payload Size setting, which are passed to the integrated block for PCI Express. The Slave Bridge can support up to eight active AXI4 memory mapped write transactions.When a remote AXI master initiates a read transaction to the slave bridge, the read address and qualifiers are captured and a MemRd request TLP is passed to the core and a completion timeout timer is started. Completions received through the core are correlated with pending read requests and read data is returned to the AXI master. The slave bridge is capable of handling up to 32 memory mapped AXI4 read requests with pending completions.

The master bridge processes both PCIe MemWr and MemRd request TLPs received from the integrated block for PCI Express and provides a means to translate addresses that are mapped within the address for PCIe domain to the memory mapped AXI4 address domain. Each PCIe MemWr request TLP header is used to create an address and qualifiers for the memory mapped AXI4 bus and the associated write data is passed to the addressed memory mapped AXI4 Slave. The master bridge can support up to eight active PCIe MemWr request TLPs.

Each PCIe MemRd request TLP header is used to create an address and qualifiers for the memory-mapped AXI4 bus. Read data is collected from the addressed memory mapped AXI4 slave and used to generate completion TLPs which are then passed to the integrated block for PCI Express. The master bridge can handle up to eight read requests with pending completions for improved AXI4 pipelining performance.

The instantiated AXI4-Stream Enhanced PCIe block contains submodules including the Requester/Completer interfaces to the AXI bridge and the Register block. The Register block contains the status, control, interrupt registers, and the AXI4-Lite interface.

Standards

The AXI Bridge for PCI Express Gen3 core is compliant with the AMBA® AXI Protocol Specification [Ref 9] and the PCI Express Base Specification v2.0 [Ref 10].

Performance and Resource Utilization

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

Port Descriptions

The interface signals for the AXI Bridge for PCI Express Gen3 are described in Table 2-1.

Table 2-1: **Top-Level Interface Signals**

Signal Name	I/O	Description
Global Signals		
refclk	I	Virtex®-7: PCIe Reference Clock. Should be driven from the O port of reference clock IBUFDS_GTE2 UltraScale™: Dynamic reconfiguration port (DRP) Clock and Internal System Clock (Half frequency from sys_clk_gt frequency). Should be driven by the ODIV2 port of reference clock IBUFDS_GTE3
sys_rst_n	I	Reset from the PCIe edge connector reset signal.
axi_aclk	O	PCIe derived clock output for M_AXI and S_AXI interfaces.
axi_ctl_aclk	I	aclk for the S_AXI_CTL interface. Recommended to be driven by the axi_aclk output.
sys_clk_gt	I	UltraScale only: PCIe Reference Clock. Should be driven from the O port of reference clock IBUFDS_GTE3.
interrupt_out	O	Interrupt signal
axi_aresetn	O	Global reset signal for AXI interfaces. During initial link-up sequence, axi_aresetn only deasserts after a function has transitioned into D0_active power state (configured and enabled).
AXI Slave Interface		
s_axi_awid[c_s_axi_id_width-1:0]	I	Slave write address ID
s_axi_awaddr[axi_addr_width-1:0]	I	Slave write address

Table 2-1: **Top-Level Interface Signals *(Cont'd)***

Signal Name	I/O	Description
s_axi_awregion[3:0]	I	Slave write region decode
s_axi_awlen[7:0]	I	Slave write burst length
s_axi_awsize[2:0]	I	Slave write burst size
s_axi_awburst[1:0]	I	Slave write burst type
s_axi_awvalid	I	Slave address write valid
s_axi_awready	O	Slave address write ready
s_axi_wdata[axi_data_width-1:0]	I	Slave write data
s_axi_wstrb[axi_data_width/8-1:0]	I	Slave write strobe
s_axi_wlast	I	Slave write last
s_axi_wvalid	I	Slave write valid
s_axi_wready	O	Slave write ready
s_axi_wuser	I	Reserved. Tie to GND.
s_axi_bid[c_s_axi_id_width-1:0]	O	Slave response ID
s_axi_bresp[1:0]	O	Slave write response
s_axi_bvalid	O	Slave write response valid
s_axi_bready	I	Slave response ready
s_axi_arid[c_s_axi_id_width-1:0]	I	Slave read address ID
s_axi_araddr[axi_addr_width-1:0]	I	Slave read address
s_axi_arregion[3:0]	I	Slave read region decode
s_axi_arlen[7:0]	I	Slave read burst length
s_axi_arsize[2:0]	I	Slave read burst size
s_axi_arburst[1:0]	I	Slave read burst type
s_axi_arvalid	I	Slave read address valid
s_axi_arready	O	Slave read address ready
s_axi_rid[c_s_axi_id_width-1:0]	O	Slave read ID tag
s_axi_rdata[axi_data_width-1:0]	O	Slave read data
s_axi_rresp[1:0]	O	Slave read response
s_axi_rlast	O	Slave read last
s_axi_rvalid	O	Slave read valid
s_axi_rready	I	Slave read ready
s_axi_ruser	O	Slave Read ECC Parity.
AXI Master Interface		
m_axi_awaddr[axi_addr_width-1:0]	O	Master write address
m_axi_awlen[7:0]	O	Master write burst length
m_axi_awsize[2:0]	O	Master write burst size

Table 2-1: **Top-Level Interface Signals *(Cont'd)***

Signal Name	I/O	Description
m_axi_awburst[1:0]	O	Master write burst type
m_axi_awprot[2:0]	O	Master write protection type
m_axi_awvalid	O	Master write address valid
m_axi_awready	I	Master write address ready
m_axi_wdata[axi_data_width-1:0]	O	Master write data
m_axi_wstrb[axi_data_width/8-1:0]	O	Master write strobe
m_axi_wlast	O	Master write last
m_axi_wvalid	O	Master write valid
m_axi_wready	I	Master write ready
m_axi_wuser	O	Reserved. Internally tied to GND.
m_axi_bresp[1:0]	I	Master write response
m_axi_bvalid	I	Master write response valid
m_axi_bready	O	Master response ready
m_axi_araddr[axi_addr_width-1:0]	O	Master read address
m_axi_arlen[7:0]	O	Master read burst length
m_axi_arsize[2:0](1)	O	Master read burst size
m_axi_arburst[1:0]	O	Master read burst type
m_axi_arprot[2:0]	O	Master read protection type
m_axi_arvalid	O	Master read address valid
m_axi_arready	I	Master read address ready. This signal only responds when Bus Master Enable bit is set in the Command register within PCI™ Configuration Space.
m_axi_rdata[axi_data_width-1:0]	I	Master read data
m_axi_rresp[1:0]	I	Master read response
m_axi_rlast	I	Master read last
m_axi_rvalid	I	Master read valid
m_axi_rready	O	Master read ready
m_axi_ruser	I	Reserved. Tie to GND.
AXI4-Lite Control Interface		
Endpoint configuration: s_axi_ctl_awaddr[11:0] Root Port configuration: s_axi_ctl_awaddr[27:0]	I	Slave write address
s_axi_ctl_awvalid	I	Slave write address valid
s_axi_ctl_awready	O	Slave write address ready
s_axi_ctl_wdata[31:0]	I	Slave write data

Table 2-1: **Top-Level Interface Signals *(Cont'd)***

Signal Name	I/O	Description
s_axi_ctl_wstrb[3:0]	I	Slave write strobe
s_axi_ctl_wvalid	I	Slave write valid
s_axi_ctl_wready	O	Slave write ready
s_axi_ctl_bresp[1:0]	O	Slave write response
s_axi_ctl_bvalid	O	Slave write response valid
s_axi_ctl_bready	I	Slave response ready
s_axi_ctl_araddr[31:0]	I	Slave read address (only one configuration at a time)
Endpoint configuration: s_axi_ctl_araddr[11:0]		
Root Port configuration: s_axi_ctl_araddr[27:0]		
s_axi_ctl_arvalid	I	Slave read address valid
s_axi_ctl_arready	O	Slave read address ready
s_axi_ctl_rdata[31:0]	O	Slave read data
s_axi_ctl_rresp[1:0]	O	Slave read response
s_axi_ctl_rvalid	O	Slave read valid
s_axi_ctl_rready	I	Slave read ready
MSI Signals		
intx_msi_request	I	Legacy interrupt input (see pf0_interrupt_pin) when msi_enable = 0. Initiates a MSI write request when msi_enable = 1. Intx_msi_request is asserted for one clock period.
intx_msi_grant	O	Indicates legacy interrupt/MSI grant signal. The intx_msi_grant signal is asserted for one clock period when the interrupt is accepted by the PCIe core.
msi_enable	O	Indicates when MSI is enabled.
msi_vector_num [4:0]	I	Indicates MSI vector to send when writing a MSI write request.
msi_vector_width [2:0]	O	Indicates the size of the MSI field (the number of MSI vectors allocated to the device).
MSI-X Signals		
cfg_interrupt_msix_enable	O	Configuration Interrupt MSI-X Function Enabled. When asserted, indicates that the Message Signaling Interrupt (MSI-X) messaging is enabled, per function.
cfg_interrupt_msix_mask	O	Configuration Interrupt MSI-X Function Mask. Indicates the state of the Function Mask bit in the MSI-X Message Control field, per function.

Table 2-1: **Top-Level Interface Signals *(Cont'd)***

Signal Name	I/O	Description
cfg_interrupt_msix_address	I	Configuration Interrupt MSI-X Address. When the core is configured to support MSI-X interrupts, this bus is used by the user logic to communicate the address to be used for an MSI-X message.
cfg_interrupt_msix_data	I	Configuration Interrupt MSI-X Data. When the core is configured to support MSI-X interrupts, this bus is used by the user logic to communicate the data to be used for an MSI-X message.
cfg_interrupt_msix_int	I	Configuration Interrupt MSI-X Data Valid. This signal indicates that valid information has been placed on the cfg_interrupt_msix_address[63:0] and cfg_interrupt_msix_data[31:0] buses, and the originating function number has been placed on cfg_interrupt_msi_function_number[3:0]. The core internally registers the associated address and data from cfg_interrupt_msix_address and cfg_interrupt_msix_data on the 0-to-1 transition of this valid signal. The user application must ensure that the cfg_interrupt_msix_enable bit corresponding to function in use is set before asserting cfg_interrupt_msix_int. After asserting an interrupt, the user logic must wait for the cfg_interrupt_msix_sent or cfg_interrupt_msix_fail indication from the core before asserting a new interrupt.
cfg_interrupt_msix_sent	O	Configuration Interrupt MSI-X Interrupt Sent. The core generates a one-cycle pulse on this output to indicate that it has accepted the information placed on the cfg_interrupt_msix_address[63:0] and cfg_interrupt_msix_data[31:0] buses, and an MSI-X interrupt message has been transmitted on the link. The user application must wait for this pulse before signaling another interrupt condition to the core.
cfg_interrupt_msix_fail	O	Configuration Interrupt MSI-X Interrupt Operation Failed. A one-cycle pulse on this output indicates that the interrupt controller has failed to transmit MSI-X interrupt on the link. The user application must retransmit the MSI-X interrupt in this case.
PCIe Interface		
pci_exp_rxp[PL_LINK_CAP_MAX_LINK_WIDTH-1:0]	I	PCIe RX serial interface
pci_exp_rxn[PL_LINK_CAP_MAX_LINK_WIDTH-1:0]	I	PCIe RX serial interface
pci_exp_txp[PL_LINK_CAP_MAX_LINK_WIDTH-1:0]	O	PCIe TX serial interface

Table 2-1: **Top-Level Interface Signals *(Cont'd)***

Signal Name	I/O	Description
pci_exp_txn[PL_LINK_CAP_MAX_LINK_WIDTH-1:0]	O	PCIe TX serial interface

Notes:

1. When a read request is received with a length that is not 1DW and is shorter than the Master AXI data width, m_axi_arsize always indicates that the requested size is equal to the Master AXI data width. The core drops the extra data when a completion packet is formed and sent back to the requester.

Bridge Parameters

Because many features in the AXI Bridge for PCI Express Gen3 core design can be parameterized, you can uniquely tailor the implementation of the core using only the resources required for the desired functionality. This approach also achieves the best possible performance with the lowest resource usage.

The parameters defined for the AXI Bridge for PCI Express Gen3 are shown in Table 2-2.

Table 2-2: **Top-Level Parameters**

Generic	Parameter Name	Description	Allowable Values	Default Value
Bridge Parameters				
	PCIE_BLK_LOCN	PCIe integrated block location within FPGA	0: X0Y0 1: X0Y1 2: X0Y2 3:X0Y3 4:X0Y4(1) 5:X0Y5(1)	0
	PL_UPSTREAM_FACING	Configures the AXI bridge for PCIe to be a Root Port or an Endpoint	TRUE: Endpoint FALSE: Root Port	TRUE
G3	C_COMP_TIMEOUT	Selects the slave bridge completion timeout counter value	0: 50 µs 1: 50 ms	0
G6	C_AXIBAR_NUM	Number of AXI address apertures that can be accessed	1: BAR_0 enabled 2: BAR_0, BAR_1 enabled 3: BAR_0, BAR_1, BAR_2 enabled 4: BAR_0 through BAR_3 enabled 5: BAR_0 through BAR_4 enabled 6: BAR_0 through BAR_5 enabled	6
G7	C_AXIBAR_0	AXI BAR_0 aperture low address	Valid AXI address(3)(4)	0x00000000_00000000

Table 2-2: **Top-Level Parameters *(Cont'd)***

Generic	Parameter Name	Description	Allowable Values	Default Value
G8	C_AXIBAR_HIGHADDR_0	AXI BAR_0 aperture high address	Valid AXI address(3)(4)	0x00000000_00000000
G10	C_AXIBAR2PCIEBAR_0	Initial address translation from an AXI BAR_0 address to a PCI Express address	Valid address for PCIe(2)	0x00000000_00000000
G11	C_AXIBAR_1	AXI BAR_1 aperture low address	Valid AXI address(3)(4)	0x00000000_00000000
G12	C_AXIBAR_HIGHADDR_1	AXI BAR_1 aperture high address	Valid AXI address(3)(4)	0x00000000_00000000
G14	C_AXIBAR2PCIEBAR_1	Initial address translation from an AXI BAR_1 address to a PCI Express address	Valid address for PCIe(2)	0x00000000_00000000
G15	C_AXIBAR_2	AXI BAR_2 aperture low address	Valid AXI address(3)(4)	0x00000000_00000000
G16	C_AXIBAR_HIGHADDR_2	AXI BAR_2 aperture high address	Valid AXI address(3)(4)	0x00000000_00000000
G18	C_AXIBAR2PCIEBAR_2	Initial address translation from an AXI BAR_2 address to a PCI Express address	Valid address for PCIe(2)	0x00000000_00000000
G19	C_AXIBAR_3	AXI BAR_3 aperture low address	Valid AXI address(3)(4)	0x00000000_00000000
G20	C_AXIBAR_HIGHADDR_3	AXI BAR_3 aperture high address	Valid AXI address(3)(4)	0x00000000_00000000
G22	C_AXIBAR2PCIEBAR_3	Initial address translation from an AXI BAR_3 address to a PCI Express address	Valid address for PCIe(2)	0x00000000_00000000
G23	C_AXIBAR_4	AXI BAR_4 aperture low address	Valid AXI address(3)(4)	0x00000000_00000000
G24	C_AXIBAR_HIGHADDR_4	AXI BAR_4 aperture high address	Valid AXI address(3)(4)	0x00000000_00000000
G26	C_AXIBAR2PCIEBAR_4	Initial address translation from an AXI BAR_4 address to a PCI Express address	Valid address for PCIe(2)	0x00000000_00000000
G27	C_AXIBAR_5	AXI BAR_5 aperture low address	Valid AXI address(3)(4)	0x00000000_00000000
G28	C_AXIBAR_HIGHADDR_5	AXI BAR_5 aperture high address	Valid AXI address(3)(4)	0x00000000_00000000
G30	C_AXIBAR2PCIEBAR_5	Initial address translation from an AXI BAR_5 address to a PCI Express address	Valid address for PCIe(2)	0x00000000_00000000

Table 2-2: **Top-Level Parameters *(Cont'd)***

Generic	Parameter Name	Description	Allowable Values	Default Value
G31	PCIEBAR_NUM	Number of address for PCIe apertures that can be accessed	1: BAR_0 enabled 2: BAR_[0:1] enabled or BAR_1 as 64-bit 3: BAR_[0-2] enabled 4: BAR_[0-3] enabled or BAR_2 as 64-bit 5: BAR_[0-4] enabled or 6: BAR_[0-5] enabled or BAR_4 as 64-bit	3
G33	PF0_BAR0_APERTURE_SIZE	Specifies the size of the PCIe BAR	0x05: 4 Kilobytes 0x06: 8 Kilobytes ... 0x0C: 512 Kilobytes 0x0D: 1 Megabyte ... 0x16: 512 Megabytes 0x17: 1 Gigabyte ... 0x1F: 256 Gigabytes	0x05
	PF0_BAR0_CONTROL	PCI express control settings for BAR0	bit 0: • 32-bit addressable = 0 • 64-bit addressable = 1 bit 1: • Prefetchable off = 0 • Prefetchable on = 1 bit 2: • IO BAR = 0 • Memory bar = 1	100
G34	C_PCIEBAR2AXIBAR_0	Initial address translation from an AXI BAR_0 address to a PCI Express address	Valid AXI address	0x00000000_00000000

Table 2-2: **Top-Level Parameters *(Cont'd)***

Generic	Parameter Name	Description	Allowable Values	Default Value
G35	PF0_BAR1_APERTURE_SIZE	Specifies the size of the PCIe BAR.	0x05: 4 Kilobytes 0x06: 8 Kilobytes ... 0x0C: 512 Kilobytes 0x0D: 1 Megabyte ... 0x16: 512 Megabytes 0x17: 1 Gigabyte ... 0x1F: 256 Gigabytes	0x05
	PF0_BAR1_CONTROL	PCI express control settings for BAR1	bit 0: • 32-bit addressable = 0 • 64-bit addressable = 1 bit 1: • Prefetchable off = 0 • Prefetchable on = 1 bit 2: • IO bar = 0 • Memory bar = 1	100
G36	C_PCIEBAR2AXIBAR_1	Initial address translation from an AXI BAR_1 address to a PCI Express address	Valid AXI address	0x00000000_00000000
G37	PF0_BAR2_APERTURE_SIZE	Specifies the size of the PCIe BAR.	0x05: 4 Kilobytes 0x06: 8 Kilobytes ... 0x0C: 512 Kilobytes 0x0D: 1 Megabyte ... 0x16: 512 Megabytes 0x17: 1 Gigabyte ... 0x1F: 256 Gigabytes	0x05

Table 2-2: **Top-Level Parameters *(Cont'd)***

Generic	Parameter Name	Description	Allowable Values	Default Value
	PF0_BAR2_CONTROL	PCI express control settings for BAR2	bit 0: • 32-bit addressable = 0 • 64-bit addressable = 1 bit 1: • Prefetchable off = 0 • Prefetchable on = 1 bit 2: • IO bar = 0 • Memory bar = 1	100
G38	C_PCIEBAR2AXIBAR_2	Initial address translation from an AXI BAR_2 address to a PCI Express address.	Valid AXI address	0x00000000_00000000
	PF0_BAR3_APERTURE_SIZE	Specifies the size of the PCIe BAR.	0x05: 4 Kilobytes 0x06: 8 Kilobytes ... 0x0C: 512 Kilobytes 0x0D: 1 Megabyte ... 0x16: 512 Megabytes 0x17: 1 Gigabyte ... 0x1F: 256 Gigabytes	0x05
	PF0_BAR3_CONTROL	PCI express control settings for BAR3	bit 0: • 32-bit addressable = 0 • 64-bit addressable = 1 bit 1: • Prefetchable off = 0 • Prefetchable on = 1 bit 2: • IO bar = 0 • Memory bar = 1	100
	C_PCIEBAR2AXIBAR_3	Initial address translation from an AXI BAR_3 address to a PCI Express address.	Valid AXI address	0x00000000_00000000

Table 2-2: **Top-Level Parameters *(Cont'd)***

Generic	Parameter Name	Description	Allowable Values	Default Value
	PF0_BAR4_APERTURE_SIZE	Specifies the size of the PCIe BAR.	0x05: 4 Kilobytes 0x06: 8 Kilobytes ... 0x0C: 512 Kilobytes 0x0D: 1 Megabyte ... 0x16: 512 Megabytes 0x17: 1 Gigabyte ... 0x1F: 256 Gigabytes	0x05
	PF0_BAR4_CONTROL	PCI express control settings for BAR4	bit 0: • 32-bit addressable = 0 • 64-bit addressable = 1 bit 1: • Prefetchable off = 0 • Prefetchable on = 1 bit 2: • IO bar = 0 • Memory bar = 1	100
	C_PCIEBAR2AXIBAR_4	Initial address translation from an AXI BAR_4 address to a PCI Express address.	Valid AXI address	0x00000000_00000000
	PF0_BAR5_APERTURE_SIZE	Specifies the size of the PCIe BAR.	0x05: 4 Kilobytes 0x06: 8 Kilobytes ... 0x0C: 512 Kilobytes 0x0D: 1 Megabyte ... 0x16: 512 Megabytes 0x17: 1 Gigabyte ... 0x1F: 256 Gigabytes	0x05

Table 2-2: **Top-Level Parameters *(Cont'd)***

Generic	Parameter Name	Description	Allowable Values	Default Value
	PF0_BAR5_CONTROL	PCI express control settings for BAR5	bit 0: • 32-bit addressable = 0 • 64-bit addressable = 1 bit 1: • Prefetchable off = 0 • Prefetchable on = 1 bit 2: • IO bar = 0 • Memory bar = 1	100
	C_PCIEBAR2AXIBAR_5	Initial address translation from an AXI BAR_5 address to a PCI Express address.	Valid AXI address	0x00000000_00000000
AXI4-Lite Parameters				
G39	C_BASEADDR	Device base address ***Note:*** When configured as an Root Port (RP), the minimum alignment granularity must be 256 MB. Bit [27:0] are used for Bus Number, Device Number, Function number.	Valid AXI address	0xFFFF_FFFF
G40	C_HIGHADDR	Device high address	Valid AXI address	0x0000_0000
Core for PCIe Configuration Parameters				
G41	PL_LINK_CAP_MAX_LINK_WIDTH	Number of PCIe Lanes	1, 2, 4, 8	1
G42	PF0_DEVICE_ID	Device ID	16-bit vector	0x0000
G43	PF0_VENDOR_ID	Vendor ID	16-bit vector	0x0000
G44	PF0_CLASS_CODE	Class Code	24-bit vector	0x00_0000
G45	PF0_REVISION_ID	Rev ID	8-bit vector	0x00
G46	C_SUBSYSTEM_ID	Subsystem ID	16-bit vector	0x0000
G47	C_SUBSYSTEM_VENDOR_ID	Subsystem Vendor ID	16-bit vector	0x0000
G49	REF_CLK_FREQ	Reference Clock input frequency. • `refclk` for Virtex®-7 devices. • `sys_clk_gt` for UltraScale™ devices.	0: 100 MHz 1: 125 MHz 2: 250 MHz	0

Table 2-2: **Top-Level Parameters *(Cont'd)***

Generic	Parameter Name	Description	Allowable Values	Default Value
G55	PL_LINK_CAP_MAX_LINK_SPEED	Maximum PCIe link speed supported	0: 2.5 GT/s 1: 5.0 GT/s	0
	TL_LEGACY_MODE_ENABLE	Selects between Legacy PCIe Endpoint Device and regular PCIe Endpoint Device	TRUE: Legacy PCIe Endpoint Device FALSE: PCIe Endpoint Device	FALSE
	CORE_CLK_FREQ	Core Clock Frequency. See CORE CLOCK Frequency for valid configurations	1: 250 MHz 2: 500 MHz	2
	PLL_TYPE	Specifies PLL being used.	0: CPLL 1: QPLL0 2: QPLL1	0
	USER_CLK_FREQ	AXI Clock Frequency	1: 62.5 MHz 2: 125 MHz 3: 250 MHz	1
	PIPE_SIM	PIPE Simulation Enable/ Disable	TRUE: Enable PIPE Simulation FALSE: Disable PIPE Simulation	FALSE
	EXT_CH_GT_DRP	Enable/Disable Transceiver DRP Ports	TRUE: EnableTransceiver Block DRP Ports FALSE: Disable Transceiver Block DRP Ports	FALSE
	PCIE3_DRP	Enable/Disable PCIe DRP Ports	TRUE: Enable PCIe Block DRP Ports FALSE: Disable PCIe Block DRP Ports	FALSE
	DEDICATE_PERST	Use the dedicated PERST routing resources	TRUE: Use dedicated PERST signal routing FALSE: Use fabric routing for PERST signal	TRUE
	SYS_RESET_POLARITY	System Reset Polarity. An Active-Low reset should be selected for designs utilizing PCIe edge connector	0: Active-Low 1: Active-High	0
	EN_TRANSCEIVER_STATUS_PORTS	Enable Additional Transceiver Control and Status Ports	False: Do not add Transceiver debug ports TRUE: Add Transceiver debug ports	FALSE
	MSI_ENABLED	Enable/Disable MSI support	TRUE: Enable MSI-Support FALSE: Disable MSI Support	TRUE

Table 2-2: **Top-Level Parameters *(Cont'd)***

Generic	Parameter Name	Description	Allowable Values	Default Value
	DEV_PORT_TYPE	Selects PCI Express Device Type	0: PCI Express Endpoint Device 1: Legacy PCI Express Endpoint Device	0
	MSIX_EN	Enable/Disable MSI-X support	TRUE: Enable MSI-X Support FALSE: Disable MSI-X Support	FALSE
	pf0_msix_cap_pba_bir	Value of Pending Bit Array BAR Indicator Register. Indicates which BAR is used for MSI-X Pending Bit Array.	0: BAR0 1: BAR1 2: BAR2 3: BAR3 4: BAR4 5: BAR5	0
	pf0_msix_cap_pba_offset	Value of Pending Bit Array Offset Register. Indicates the address offset within the BAR where MSI-X Pending Bit Array starts.	'h0000_0000 to 'h1FFF_FFFF	'h0000_0000
	pf0_msix_cap_table_bir	Value of MSI-X Table BAR Indicator Register. Indicates which BAR is used for MSI-X table.	0: BAR0 1: BAR1 2: BAR2 3: BAR3 4: BAR4 5: BAR5	0
	pf0_msix_cap_table_offset	Value of MSI-X Table Offset Register. Indicates the address offset within the BAR where MSI-X table starts.	'h0000_0000 to 'h1FFF_FFFF	'h0000_0000
	pf0_msix_cap_table_size	Value of MSI-X Table Size Register. Indicates the size of MSI-X table.	'h000 to 'h7FF	'h000
Memory Mapped AXI4 Parameters				
G50	AXI_DATA_WIDTH	AXI Bus Data width	64 128 256	64
G51	AXI_ADDR_WIDTH	AXI Bus Address width	32-64	32
G52	C_S_AXI_ID_WIDTH	AXI Slave Bus ID width	4	4
G56	PF0_INTERRUPT_PIN	Legacy INTX pin support/ select	0: No INTX support (setting for Root Port). 1: INTA selected (only allowable when core in Endpoint configuration).	0

Table 2-2: **Top-Level Parameters *(Cont'd)***

Generic	Parameter Name	Description	Allowable Values	Default Value
AXI4 Slave Interconnect Parameters(5)				
G57	C_S_AXI_NUM_WRITE	AXI Interconnect Slave Port Write Pipeline Depth	8 Size of pipeline for active AXI AWADDR values to be stored in AXI slave bridge PCIe.	8
G58	C_S_AXI_NUM_READ	AXI Interconnect Slave Port Read Pipeline Depth	8 Size of pipeline for active AXI ARADDR values to be stored in AXI slave bridge PCIe.	8
	EN_AXI_SLAVE_IF	AXI4 Slave Bus Enable/ Disable	TRUE: AXI4 Slave Bus is active FALSE: AXI4 Slave Bus is disabled	TRUE
AXI4 Master Interconnect Parameters(5)				
G59	C_M_AXI_NUM_WRITE	AXI Interconnect master bridge write address issue depth	8 Number of actively issued AXI AWADDR values on the AXI Interconnect to the target slave device(s).	8
G60	C_M_AXI_NUM_READ	AXI Interconnect master bridge read address issue depth	8 Number of actively issued AXI ARADDR values on the AXI Interconnect to the target slave device(s).	8
	EN_AXI_MASTER_IF	AXI4 Master Bus Enable/ Disable	TRUE: AXI4 Master Bus is active FALSE: AXI4 Master Bus is disabled	TRUE

Notes:

1. X0Y4 and X0Y5 are only available on select UltraScale devices.
2. The width of this should match the address width of the PCIe BARs.
3. The range specified must comprise a complete, contiguous power of two range, such that the range = 2^n and the *n* least significant bits of the Base Address are zero.
4. The difference between C_AXIBAR_n and C_AXIBAR_HIGHADDR_n must be greater than or equal to 0x00000000_00000FFF.
5. These are the user parameters of the AXI4 Interconnect. By default, the slave bridge handles up to two AXI4 write requests and eight AXI4 read requests. The master bridge handles up to four PCIe write/read requests.

Reference Clock for PCIe Frequency Value

The reference clock input used by the serial transceiver for PCIe must be 100 MHz, 125 MHz, or 250 MHz. Note that the reference clock is `refclk` in Virtex-7 devices, and `sys_clk_gt` in UltraScale devices. The REF_CLK_FREQ parameter is used to set this value, as defined in Table 2-2. The reference clock input must be fed in from a clock source that is external to the chip.

Memory Map

The memory map shown in Table 2-3 shows the address mapping for the AXI Bridge for PCI Express Gen3 core. These registers are described in more detail in the following section. All registers are accessed through the AXI4-Lite Control Interface and are offset from C_BASEADDR. During a reset, all registers return to default values.

Table 2-3: **Register Memory Map**

Accessibility	Offset	Contents	Location
RO - EP	0x000 - 0x12F	PCIe Configuration Space Header	Part of integrated PCIe configuration space.
RO	0x130	Bridge Info	AXI bridge defined memory-mapped register space.
RO - EP	0x134	Bridge Status and Control	
R/W	0x138	Interrupt Decode	
R/W	0x13C	Interrupt Mask	
RO - EP	0x140	Bus Location	
RO	0x144	Physical-Side Interface (PHY) Status/Control	
RO - EP, R/W - RC	0x148	Root Port Status/Control	
RO - EP, R/W - RC	0x14C	Root Port MSI Base 1	
RO - EP, R/W - RC	0x150	Root Port MSI Base 2	
RO - EP, R/W - RC	0x154	Root Port Error FIFO Read	
RO - EP, R/W - RC	0x158	Root Port Interrupt FIFO Read 1	
RO - EP, R/W - RC	0x15C	Root Port Interrupt FIFO Read 2	
RO	0x160 - 0x164	Reserved (zeros returned on read)	
R/W	0x168	Configuration Control	AXI bridge defined memory-mapped register space.
RO	0x16C - 0x1FF	Reserved (zeros returned on read)	
RO	0x200	VSEC Capability 2	
RO	0x204	VSEC Header 2	
R/W	0x208 - 0x234	AXI Base Address Translation Configuration Registers	AXI bridge defined memory-mapped space.
RO	0x238 - 0xFFF	Reserved (zeros returned on read)	

PCIe Configuration Space Header

The PCIe Configuration Space Header is a memory aperture for accessing the core for PCIe configuration space. This area is read-only when configured as an Endpoint.Writes are permitted for some registers when configured as a Root Port. Special access modes can be enabled using the PHY Status/Control register. All reserved or undefined memory-mapped addresses must return zero and writes have no effect.

Bridge Info Register (Offset 0x130)

The Bridge Info register (described in Table 2-4) provides general configuration information about the AXI4-Stream Bridge. Information in this register is static and does not change during operation.

Table 2-4: **Bridge Info Register**

Bits	Name	Core Access	Reset Value	Description
0	Gen2 Capable	RO	0	If set, underlying integrated block supports PCIe Gen2 speed.
1	Root Port Present	RO	0	Indicates the underlying integrated block is a Root Port when this bit is set. If set, Root Port registers are present in this interface.
2	Up Config Capable	RO		Indicates the underlying integrated block is upconfig capable when this bit is set.
3	Gen3 Capable	RO	0	If set, underlying integrated block supports PCIe Gen3 speed.
31:4	Reserved	RO	0	Reserved

Bridge Status and Control Register (Offset 0x134)

The Bridge Status and Control register (described in Table 2-5) provides information about the current state of the AXI4-Stream Bridge. It also provides control over how reads and writes to the Core Configuration Access aperture are handled.

Table 2-5: **Bridge Status and Control Register**

Bits	Name	Core Access	Reset Value	Description
7:0	Reserved	RO	0	Reserved
8	Global Disable	RW	0	When set, disables interrupt line from being asserted. Does not prevent bits in Interrupt Decode register from being set.
31:9	Reserved	RO	0	Reserved

Interrupt Decode Register (Offset 0x138)

The Interrupt Decode register (described in Table 2-6) provides a single location where the host processor interrupt service routine can determine what is causing the interrupt to be asserted and how to clear the interrupt. Writing a `1'b1` to any bit of the Interrupt Decode register clears that bit except for the Correctable, Non-Fatal, and Fatal bits.

Follow this sequence to clear the Correctable, Non-Fatal, and Fatal bits:

1. Clear the Root Port Error FIFO (0x154) by performing first a read, followed by write-back of the same register.
2. Read Root Port Status/Control Register (0x148) bit 16, and ensure that the Error FIFO is empty.

   Note: If the error FIFO is still not empty, repeat step 1 and step 2 until the Error FIFO is empty.
3. Write to the Interrupt Decode Register (0x138) with 1 to the appropriate error bit to clear it.

IMPORTANT: *An asserted bit in the Interrupt Decode register does not cause the interrupt line to assert unless the corresponding bit in the Interrupt Mask register is also set.*

Table 2-6: **Interrupt Decode Register**

Bits	Name	Core Access	Reset Value	Description
0	Link Down	RW1C	0	Indicates that Link-Up on the PCI Express link was lost. Not asserted unless link-up had previously been seen.
1	Reserved	RO	0	Reserved
2	Streaming Error	RW1C	0	Indicates a gap was encountered in a streamed packet on the TX interface (RW, RR, or CC).
3	Hot Reset	RW1C	0	Indicates a Hot Reset was detected.
4	Reserved	RO	0	Reserved
7:5	Cfg Completion Status	RW1C	0	Indicates config completion status.
8	Cfg Timeout	RW1C	0	Indicates timeout on an enhanced configuration access mechanism (ECAM) access. (Only applicable to Root Port cores.)
9	Correctable	RW1C	0	Indicates a correctable error message was received. Requester ID of error message should be read from the Root Port FIFO. (Only applicable to Root Port cores.)
10	Non-Fatal	RW1C	0	Indicates a non-fatal error message was received. Requester ID of error message should be read from the Root Port FIFO. (Only applicable to Root Port cores.)

Table 2-6: **Interrupt Decode Register *(Cont'd)***

Bits	Name	Core Access	Reset Value	Description
11	Fatal	RW1C	0	Indicates a fatal error message was received. Requester ID of error message should be read from the Root Port FIFO. (Only applicable to Root Port cores.)
15:12	Reserved	RO	0	Reserved
16	INTx Interrupt Received	RW1C	0	Indicates an INTx interrupt was received. Interrupt details should be read from the Root Port FIFO. (Only applicable to Root Port cores.)
17	MSI Interrupt Received	RW1C	0	Indicates an MSI(x) interrupt was received. Interrupt details should be read from the Root Port FIFO. (Only applicable to Root Port cores.)
19:18	Reserved	RO	0	Reserved
20	Slave Unsupported Request	RW1C	0	Indicates that a completion TLP was received with a status of `0b001` - Unsupported Request.
21	Slave Unexpected Completion	RW1C	0	Indicates that a completion TLP was received that was unexpected.
22	Slave Completion Timeout	RW1C	0	Indicates that the expected completion TLP(s) for a read request for PCIe was not returned within the time period selected by the C_COMP_TIMEOUT parameter.
23	Slave Error Poison	RW1C	0	Indicates the error poison (EP) bit was set in a completion TLP.
24	Slave Completer Abort	RW1C	0	Indicates that a completion TLP was received with a status of 0b100 - Completer Abort.
25	Slave Illegal Burst	RW1C	0	Indicates that a burst type other than INCR was requested by the AXI master.
26	Master DECERR	RW1C	0	Indicates a Decoder Error (DECERR) response was received.
27	Master SLVERR	RW1C	0	Indicates a Slave Error (SLVERR) response was received.
28	Master Error Poison	RW1C	0	Indicates an EP bit was set in a MemWR TLP for PCIe.
31:29	Reserved	RO	0	Reserved

Interrupt Mask Register (Offset 0x13C)

The Interrupt Mask register controls whether each individual interrupt source can cause the interrupt line to be asserted. A one in any location allows the interrupt source to assert the interrupt line. The Interrupt Mask register initializes to all zeros. Therefore, by default no interrupt is generated for any event. Table 2-7 describes the Interrupt Mask register bits and values.

Table 2-7: **Interrupt Mask Register**

Bits	Name	Core Access	Reset Value	Description
0	Link Down	RW	0	Enables interrupts for Link Down events when bit is set.
1	Reserved	RO	0	Reserved
2	Streaming Error	RW	0	Enables interrupts for Streaming Error events when bit is set.
3	Hot Reset	RW	0	Enables interrupts for Hot Reset events when bit is set. (Only writable for EP configurations, otherwise = 0)
4	Reserved	RO	0	Reserved
7:5	Cfg Completion Status	RW	0	Enables interrupts for config completion status. (Only writable for Root Port Configurations, otherwise = 0)
8	Cfg Timeout	RO	0	Enables interrupts for Config (Cfg) Timeout events when bit is set. (Only writable for Root Port Configurations, otherwise = 0)
9	Correctable	RO	0	Enables interrupts for Correctable Error events when bit is set. (Only writable for Root Port Configurations, otherwise = 0)
10	Non-Fatal	RO	0	Enables interrupts for Non-Fatal Error events when bit is set. (Only writable for Root Port Configurations, otherwise = 0)
11	Fatal	RO	0	Enables interrupts for Fatal Error events when bit is set. (Only writable for Root Port Configurations, otherwise = 0)
15:12	Reserved	RO	0	Reserved
16	INTx Interrupt Received	RO	0	Enables interrupts for INTx Interrupt events when bit is set. (Only writable for Root Port Configurations, otherwise =0)
17	MSI Interrupt Received	RO	0	Enables interrupts for MSI Interrupt events when bit is set. (Only writable for Root Port Configurations, otherwise =0)
19:18	Reserved	RO	0	Reserved
20	Slave Unsupported Request	RW	0	Enables the Slave Unsupported Request interrupt when bit is set.
21	Slave Unexpected Completion	RW	0	Enables the Slave Unexpected Completion interrupt when bit is set.
22	Slave Completion Timeout	RW	0	Enables the Slave Completion Timeout interrupt when bit is set.
23	Slave Error Poison	RW	0	Enables the Slave Error Poison interrupt when bit is set.
24	Slave Completer Abort	RW	0	Enables the Slave Completer Abort interrupt when bit is set.
25	Slave Illegal Burst	RW	0	Enables the Slave Illegal Burst interrupt when bit is set.
26	Master DECERR	RW	0	Enables the Master DECERR interrupt when bit is set.
27	Master SLVERR	RW	0	Enables the Master SLVERR interrupt when bit is set.
28	Master Error Poison	RW	0	Enables the Master Error Poison interrupt when bit is set.
31:29	Reserved	RO	0	Reserved

Bus Location Register (Offset 0x140)

The Bus Location register reports the Bus, Device, and Function number, and the Port number for the PCIe port (Table 2-8).

Table 2-8: **Bus Location Register**

Bits	Name	Core Access	Reset Value	Description
2:0	Function Number	RO	0	Function number of the port for PCIe. Hard-wired to 0.
7:3	Device Number	RO	0	Device number of port for PCIe. For Endpoint, this register is RO and is set by the Root Port.
15:8	Bus Number	RO	0	Bus number of port for PCIe. For Endpoint, this register is RO and is set by the external Root Port.
23:16	Port Number	RW	0	Sets the Port number field of the Link Capabilities register. EP: Always Read 0 and is not writeable. RP: Is writeable.
31:24	Reserved	RO	0	Reserved

PHY Status/Control Register (Offset 0x144)

The PHY Status/Control register (described in Table 2-9) provides the status of the current PHY state, as well as control of speed and rate switching for Gen2-capable cores.

Table 2-9: **PHY Status/Control Register**

Bits	Name	Core Access	Reset Value	Description
0	Link Rate is Gen2	RO	0	Reports the current link rate. • 0b = 2.5 GT/s (if bit[12] = 0), or 8.0GT/s (if bit[12] = 1) • 1b = 5.0 GT/s
2:1	Link Width	RO	0	Reports the current link width. 00b = x1, 01b = x2, 10b = x4, 11b = x8.
8:3	LTSSM State	RO	0	Reports the current Link Training and Status State Machine (LTSSM) state. Encoding is specific to the underlying integrated block.
10:9	Reserved	RO	0	• Reserved
11	Link Up	RO	0	Reports the current PHY Link-up state. • 1b: Link up • 0b: Link down Link up indicates the core has achieved link up status, meaning the LTSSM is in the L0 state and the core can send/receive data packets.
12	Link Rate is Gen3	RO	0	Reports the current link rate. 0b = see bit[0]. 1b = 8.0 GT/s
31:13	Reserved	RO	0	Reserved

Root Port Status/Control Register (Offset 0x148)

The Root Port Status/Control register provides access to the Root Port specific status and control. This register is only implemented for Root Port cores. For non-Root Port cores, reads return 0 and writes are ignored (described in Table 2-10).

Table 2-10: **Root Port Status/Control Register**

Bits	Name	Core Access	Reset Value	Description
0	Bridge Enable	RW	0	When set, allows the reads and writes to the AXIBARs to be presented on the PCIe bus. Root Port software needs to write a 1 to this bit when enumeration is done. AXI Enhanced PCIe Bridge clears this location when link up to link down transition occurs. Default is set to 0.
15:1	Reserved	RO	0	Reserved.
16	Error FIFO Not Empty	RO	0	Indicates that the Root Port Error FIFO has data to read.
17	Error FIFO Overflow	RW1C	0	Indicates that the Root Port Error FIFO overflowed and an error message was dropped. Writing a 1 clears the overflow status.
18	Interrupt FIFO Not Empty	RO	0	Indicates that the Root Port Interrupt FIFO has data to read.
19	Interrupt FIFO Overflow	RW1C	0	Indicates that the Root Port Interrupt FIFO overflowed and an interrupt message was dropped. Writing a 1 clears the overflow status
31:20	Reserved	RO	0	Reserved.

Root Port MSI Base Register 1 (Offset 0x14C)

The Root Port MSI Base register contains the upper 32-bits of the 64-bit MSI address (described in Table 2-11).

For EP configurations, read returns zero.

Table 2-11: **Root Port MSI Base Register 1**

Bits	Name	Core Access	Reset Value	Description
31:0	MSI Base	RW	0	4Kb-aligned address for MSI interrupts. In case of 32-bit MSI, it returns 0 but captures the upper 32-bits of the MSI address in case of 64-bit MSI.

Root Port MSI Base Register 2 (Offset 0x150)

The Root Port MSI Base register 2 (described in Table 2-12) sets the address window in Root Port cores used for MSI interrupts. MemWr TLPs to addresses in this range are interpreted as MSI interrupts. MSI TLPs are interpreted based on the address programmed in this

register. The window is always 4 Kb, beginning at the address indicated in this register. For EP configurations, a read returns zero. However, the AXI Bridge for PCI Express Gen3 core does not support MSI-X and multiple vector address, only single MSI is supported.

Table 2-12: **Root Port MSI Base Register 2**

Bits	Name	Core Access	Reset Value	Description
11:0	Reserved	RO	0	Reserved
31:12	MSI Base	RW	0	4 Kb-aligned address for MSI interrupts.

Root Port Error FIFO Read Register (Offset 0x154)

Reads from this location return a queued error (Correctable/Non-fatal/Fatal) messages. Data from each read follows the format shown in Table 2-13. For EP configurations, read returns zero.

Reads are non-destructive. Removing the message from the FIFO requires a write. The write value is ignored.

Table 2-13: **Root Port Error FIFO Read Register**

Bits	Name	Core Access	Reset Value	Description
15:0	Requester ID	RWC	0	Requester ID belonging to the requester of the error message.
17:16	Error Type	RWC	0	Indicates the type of the error. 00b: Correctable 01b: Non-Fatal 10b: Fatal 11b: Reserved
18	Error Valid	RWC	0	Indicates whether read succeeded. 1b: Success 0b: No message to read
31:19	Reserved	RO	0	Reserved

Root Port Interrupt FIFO Read Register 1 (Offset 0x158)

Reads from this location return queued interrupt messages. Data from each read follows the format shown in Table 2-14. For MSI interrupts, the message payload is presented in the Root Port Interrupt FIFO Read 2 register. The interrupt-handling flow should be to read this register first, immediately followed by the Root Port Interrupt FIFO Read 2 register. For non-Root Port cores, reads return zero.

Note: Reads are non-destructive. Removing the message from the FIFO requires a write to either this register or the Root Port Interrupt FIFO Read 2 register. The write value is ignored.

Table 2-14: **Root Port Interrupt FIFO Read Register 1**

Bits	Name	Core Access	Reset Value	Description
15:0	Requester ID	RWC	0	Requester ID belonging to the requester of the error message.
26:16	MSI Address	RWC	0	For MSI interrupts, contains address bits 12:2 from the TLP address field.
28:27	Interrupt Line	RWC	0	Indicates interrupt line used. 00b: INTA 01b: INTB 10b: INTC 11b: INTD For MSI, this field is set to 00b and should be ignored.
29	Interrupt Assert	RWC	0	Indicates assert or deassert for INTx. 1b: Assert 0b: Deassert For MSI, this field is set to 0b and should be ignored.
30	MSI Interrupt	RWC	0	Indicates whether interrupt is MSI or INTx. 1b = MSI, 0b = INTx.
31	Interrupt Valid	RWC	0	Indicates whether read succeeded. 1b: Success 0b: No interrupt to read

Root Port Interrupt FIFO Read Register 2 (Offset 0x15C)

Reads from this location return queued interrupt messages. Data from each read follows the format shown in Table 2-15. For MSI interrupts, the message payload is presented in this register, while the header information is presented in the Root Port Interrupt FIFO Read 1 register. The interrupt-handling flow should be to read the Root Port Interrupt FIFO Read 1 register first, immediately followed by this register. For non-Root Port cores, reads return 0. For INTx interrupts, reads return zero.

Note: Reads are non-destructive. Removing the message from the FIFO requires a write to either this register or the Root Port Interrupt FIFO Read 1 register (write value is ignored).

Table 2-15: **Root Port Interrupt FIFO Read Register 2**

Bits	Name	Core Access	Reset Value	Description
15:0	Message Data	RWC	0	Payload for MSI messages.
31:16	Reserved	RO	0	Reserved

Configuration Control Register (Offset 0x168)

Configuration Control register (described in Table 2-16) allows the user application to indicate if a correctable or uncorrectable error has occurred and report it in the respective AER Error Status Register.

Table 2-16: **Configuration Control Register**

Bits	Name	Core Access	Reset Value	Description
0	Uncorrectable Error	RW	0	Uncorrectable Error Detected. The user application writes a 1 to this bit to indicate an Uncorrectable error was detected within the user logic that needs to be reported as an internal error through the PCI Express Advanced Error Reporting mechanism. In response, the core sets the Uncorrected Internal Error Status bit in the AER Uncorrectable Error Status Register of all enabled functions, and also sends an error message if enabled to do so. This error is not considered function-specific. This bit only asserts for 1 clock cycle and automatically resets to 0 in the next clock cycle.
1	Correctable Error	RW	0	Correctable Error Detected. The user application writes a 1 to this bit to indicate a Correctable error was detected within the user logic that needs to be reported as an internal error through the PCI Express Advanced Error Reporting mechanism. In response, the core sets the Corrected Internal Error Status bit in the AER Correctable Error Status Register of all enabled functions, and also sends an error message if enabled to do so. This error is not considered function-specific. This bit only asserts for 1 clock cycle and automatically resets to 0 in the next clock cycle.
31:2	Reserved	RO	0	Reserved

VSEC Capability Register 2 (Offset 0x200)

The VSEC capability register (described in Table 2-17) allows the memory space for the core to appear as though it is a part of the underlying integrated block PCIe configuration space. The VSEC is inserted immediately following the last enhanced capability structure in the underlying block. VSEC is defined in §7.18 of the PCI Express Base Specification, v1.1 (§7.19 of v2.0) [Ref 10].

This register is only included if C_INCLUDE_BAR_OFFSET_REG = 1.

Table 2-17: **VSEC Capability Register 2**

Bits	Name	Core Access	Reset Value	Description
15:0	VSEC Capability ID	RO	0x000B	PCI-SIG™ defined ID identifying this Enhanced Capability as a Vendor-Specific capability. Hardcoded to 0x000B.
19:16	Capability Version	RO	0x1	Version of this capability structure. Hardcoded to 0x1.
31:20	Next Capability Offset	RO	0x000	Offset to next capability.

VSEC Header Register 2 (Offset 0x204)

The VSEC Header Register 2 (described in Table 2-18) provides a unique (within a given vendor) identifier for the layout and contents of the VSEC structure, as well as its revision and length. VSEC Header Register 2 is part of the AXI Bridge for PCI Express Gen3 core that contains AXI Base Address Translation Configuration Registers which start immediately after VSEC Header Register 2 (Offset 0x208).

This register is only included if C_INCLUDE_BAR_OFFSET_REG = 1.

Table 2-18: **VSEC Header Register 2**

Bits	Name	Core Access	Reset Value	Description
15:0	VSEC ID	RO	0x0002	ID value uniquely identifying the nature and format of this VSEC structure.
19:16	VSEC REV	RO	0x0	Version of this capability structure. Hardcoded to 0x0.
31:20	VSEC Length	RO	0x038	Length of the entire VSEC Capability structure, in bytes, including the VSEC Capability register. Hardcoded to 0x038 (56 decimal).

AXI Base Address Translation Configuration Registers (Offset 0x208 - 0x234)

The AXI Base Address Translation Configuration registers and their offsets are shown in Table 2-19 and the register bits are described in Table 2-20. This set of registers can be used in two configurations based on the address width of the PCIe BARs. When the PCIe BAR is set to a 32-bit address space, then the translation vector should be placed into the AXIBAR2PCIEBAR_nL register where n is the PCIe BAR number. When the BAR is set to a 64-bit address space, then the most significant 32 bits are written into the AXIBAR2PCIEBAR_nU and the least significant 32 bits are written into AXIBAR2PCIEBAR_nL. These registers are only included if C_INCLUDE_BAR_OFFSET_REG = 1. Care should be taken so that invalid values are not written to the address translation registers.

Table 2-19: **AXI Base Address Translation Configuration Registers**

Offset	Bits	Register Mnemonic
0x208	31-0	AXIBAR2PCIEBAR_0U
0x20C	31-0	AXIBAR2PCIEBAR_0L
0x210	31-0	AXIBAR2PCIEBAR_1U
0x214	31-0	AXIBAR2PCIEBAR_1L
0x218	31-0	AXIBAR2PCIEBAR_2U
0x21C	31-0	AXIBAR2PCIEBAR_2L
0x220	31-0	AXIBAR2PCIEBAR_3U
0x224	31-0	AXIBAR2PCIEBAR_3L
0x228	31-0	AXIBAR2PCIEBAR_4U
0x22C	31-0	AXIBAR2PCIEBAR_4L
0x230	31-0	AXIBAR2PCIEBAR_5U
0x234	31-0	AXIBAR2PCIEBAR_5L

Table 2-20: **AXI Base Address Translation Configuration Register Bit Definitions**

Bits	Name	Core Access	Reset Value	Description
31-0	Lower Address	R/W	C_AXIBAR2PCIEBAR_0(31 to 0)	To create the address for PCIe–this is the value substituted for the least significant 32 bits of the AXI address.

31-0	Upper Address	R/W	if (C_AXIBAR2PCIEBAR_0 = 64 bits), then reset value = C_AXIBAR2PCIEBAR_0(63 to 32)
			
			if (C_AXIBAR2PCIEBAR_0 = 32 bits), then reset value = 0x00000000
				To create the address for PCIe–this is the value substituted for the most significant 32 bits of the AXI address.

31-0	Lower Address	R/W	C_AXIBAR2PCIEBAR_1(31 to 0)	To create the address for PCIe–this is the value substituted for the least significant 32 bits of the AXI address.

31-0	Upper Address	R/W	if (C_AXIBAR2PCIEBAR_1 = 64 bits), then reset value = C_AXIBAR2PCIEBAR_1(63 to 32)
			
			if (C_AXIBAR2PCIEBAR_1 = 32 bits), then reset value = 0x00000000
				To create the address for PCIe– this is the value substituted for the most significant 32 bits of the AXI address.

31-0	Lower Address	R/W	C_AXIBAR2PCIEBAR_2(31 to 0)	To create the address for PCIe–this is the value substituted for the least significant 32 bits of the AXI address.

Table 2-20: **AXI Base Address Translation Configuration Register Bit Definitions *(Cont'd)***

Bits	Name	Core Access	Reset Value	Description

31-0	Upper Address	R/W	if (C_AXIBAR2PCIEBAR_2 = 64 bits), then reset value = C_AXIBAR2PCIEBAR_2(63 to 32)
			
			if (C_AXIBAR2PCIEBAR_2 = 32 bits), then reset value = 0x00000000
				To create the address for PCIe–this is the value substituted for the most significant 32 bits of the AXI address.

31-0	Lower Address	R/W	C_AXIBAR2PCIEBAR_3(31 to 0)	To create the address for PCIe–this is the value substituted for the least significant 32 bits of the AXI address.

31-0	Upper Address	R/W	if (C_AXIBAR2PCIEBAR_3 = 64 bits) then reset value = C_AXIBAR2PCIEBAR_3(63 to 32)
			
			if (C_AXIBAR2PCIEBAR_3 = 32 bits) then reset value = 0x00000000
				To create the address for PCIe–this is the value substituted for the most significant 32 bits of the AXI address.

31-0	Lower Address	R/W	C_AXIBAR2PCIEBAR_4(31 to 0)	To create the address for PCIe–this is the value substituted for the least significant 32 bits of the AXI address.

31-0	Upper Address	R/W	if (C_AXIBAR2PCIEBAR_4 = 64 bits), then reset value = C_AXIBAR2PCIEBAR_4(63 to 32)
			
			if (C_AXIBAR2PCIEBAR_4 = 32 bits), then reset value = 0x00000000
				To create the address for PCIe–this is the value substituted for the most significant 32 bits of the AXI address.

31-0	Lower Address	R/W	C_AXIBAR2PCIEBAR_5(31 to 0)	To create the address for PCIe–this is the value substituted for the least significant 32 bits of the AXI address.

31-0	Upper Address	R/W	if (C_AXIBAR2PCIEBAR_5 = 64 bits), then reset value = C_AXIBAR2PCIEBAR_5(63 to 32)
			
			if (C_AXIBAR2PCIEBAR_5 = 32 bits), then reset value = 0x00000000
				To create the address for PCIe–this is the value substituted for the most significant 32 bits of the AXI address.

Enhanced Configuration Access

When the AXI Bridge for PCI Express Gen3 core is configured as a Root Port, configuration traffic is generated by using the PCI Express enhanced configuration access mechanism (ECAM). ECAM functionality is available only when the core is configured as a Root Port. Reads and writes to a certain memory aperture are translated to configuration reads and writes, as specified in the PCI Express Base Specification (v3.0), §7.2.2 [Ref 10].

The address breakdown is defined in Table 2-21.

When an ECAM access is attempted to the primary bus number, which defaults as bus 0 from reset, then access to the type 1 PCI™ Configuration Header of the integrated block in the Enhanced Interface for PCIe is performed. When an ECAM access is attempted to the secondary bus number, then type 0 configuration transactions are generated. When an ECAM access is attempted to a bus number that is in the range defined by the secondary bus number and subordinate bus number (not including the secondary bus number), then type 1 configuration transactions are generated. The primary, secondary, and subordinate bus numbers are written by Root Port software to the type 1 PCI Configuration Header of the Enhanced Interface for PCIe in the beginning of the enumeration procedure.

When an ECAM access is attempted to a bus number that is out of the bus_number and subordinate bus number range, the bridge does not generate a configuration request and signal SLVERR response on the AXI4-Lite bus. When the AXI Bridge for PCI Express Gen3 is configured for EP (PL_UPSTREAM_FACING = TRUE), the underlying Integrated Block configuration space and the core memory map are available at the beginning of the memory space. The memory space looks like a simple PCI Express configuration space. When the AXI Bridge for PCI Express Gen3 is configured for RC (PL_UPSTREAM_FACING = FALSE), the same is true, but it also looks like an ECAM access to primary bus, Device 0, Function 0.

When the AXI Bridge for PCI Express Gen3 core is configured as a Root Port, the reads and writes of the local ECAM are Bus 0. Because the FPGA only has a single Integrated Block for PCIe core, all local ECAM operations to Bus 0 return the ECAM data for Device 0, Function 0.

Configuration write accesses across the PCI Express bus are non-posted writes and block the AXI4-Lite interface while they are in progress. Because of this, system software is not able to service an interrupt if one were to occur. However, interrupts due to abnormal terminations of configuration transactions can generate interrupts. ECAM read transactions block subsequent Requester read TLPs until the configuration read completions packet is returned to allow unique identification of the completion packet.

Table 2-21: **ECAM Addressing**

Bits	Name	Description
1:0	Byte Address	Ignored for this implementation. The S_AXI_CTL_WSTRB[3:0] signals define byte enables for ECAM accesses.
7:2	Register Number	Register within the configuration space to access.
11:8	Extended Register Number	Along with Register Number, allows access to PCI Express Extended Configuration Space.
14:12	Function Number	Function Number to completer.
19:15	Device Number	Device Number to completer.
27:20	Bus Number	Bus Number to completer.

Designing with the Core

This chapter includes guidelines and additional information to make designing with the core easier.

General Design Guidelines

The Xilinx® Vivado® Design Suite has been optimized to provide a starting point for designing with the AXI Bridge for PCI Express Gen3 core.

Clocking

The reference clock input is used to generate the internal clocks used by the core and the output clock. Note that the reference clock is `refclk` in Virtex®-7 devices, and `sys_clk_gt` in UltraScale™ devices. This clock must be provided at the reference clock frequency selected in the Vivado Integrated Design Environment (IDE) during IP generation. This port should be driven by the PCI Express edge connector clock pins through an IBUFDSGTE primitive.

The `axi_aclk` output is the clock used for all AXI interfaces and should drive all corresponding AXI Interconnect `aclk` signals as well as the `axi_ctl_aclk` input port.

RECOMMENDED: *To facilitate timing, `axi_aclk` output should not be used for the system clock for your design.*

For additional information about how the source of the `aclk` clock might impact your designs, see the *7 Series FPGAs Clocking Resources User Guide* (UG472) [Ref 7], or the *UltraScale Architecture Clocking Resources User Guide* (UG572) [Ref 8].

Figure 3-1 shows the clocking diagram for the core in an UltraScale device.

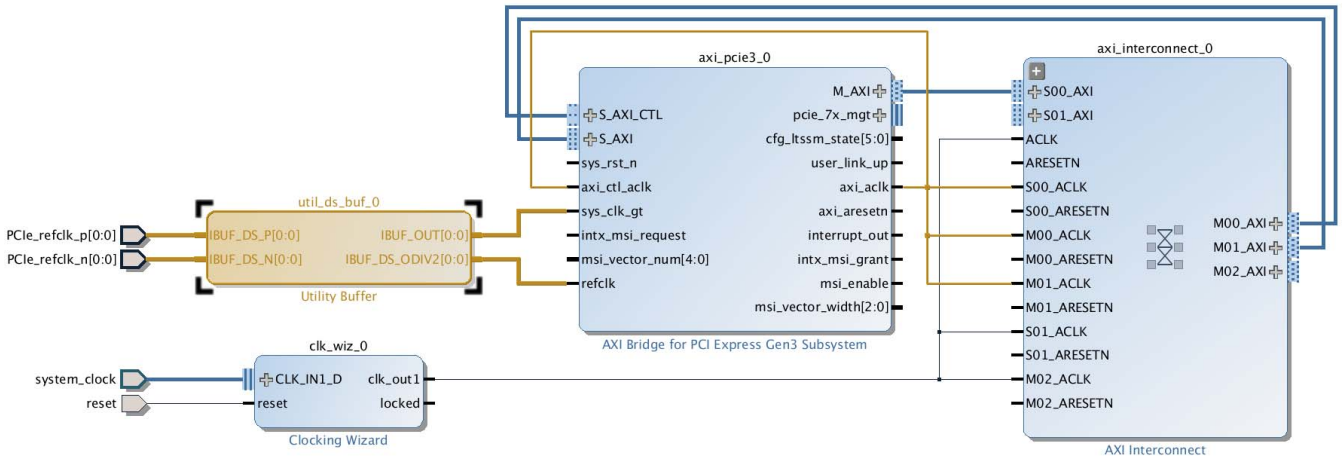

Figure 3-1: **Clocking Diagram**

Resets

For endpoint configurations, the `sys_rst_n` signal should be driven by the PCI Express edge connector reset (`perstn`). This serves as the reset for the PCI Express interface.

The `axi_aresetn` output the AXI reset signal synchronous with the clock provided on the `axi_aclk` output. This reset should drive all corresponding AXI Interconnect `aresetn` signals.

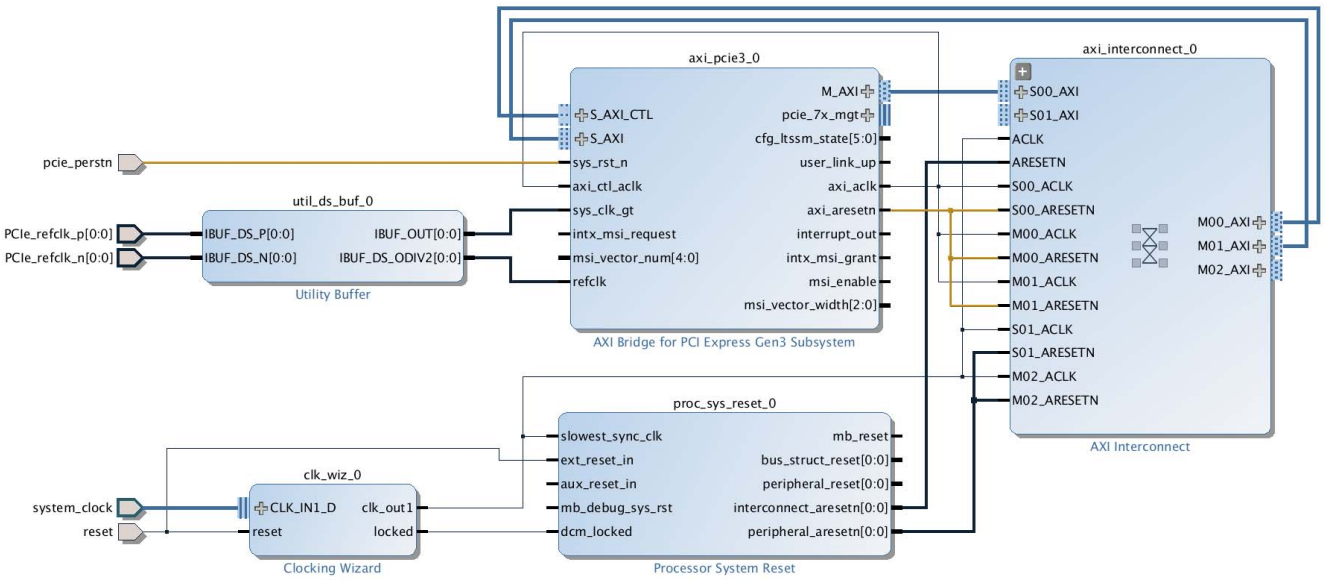

Figure 3-2: **System Reset Connection**

AXI Transactions for PCIe

Table 3-1 and Table 3-2 are the translation tables for AXI4-Stream and memory-mapped transactions.

Table 3-1: **AXI4 Memory-Mapped Transactions to AXI4-Stream PCIe TLPs**

AXI4 Memory-Mapped Transaction	AXI4-Stream PCIe TLPs
INCR Burst Read of AXIBAR	MemRd 32 (3DW)
INCR Burst Write to AXIBAR	MemWr 32 (3DW)
INCR Burst Read of AXIBAR	MemRd 64 (4DW)
INCR Burst Write to AXIBAR	MemWr 64 (4DW)

Table 3-2: **AXI4-Stream PCIe TLPs to AXI4 Memory Mapped Transactions**

AXI4-Stream PCIe TLPs	AXI4 Memory-Mapped Transaction
MemRd 32 (3DW) of PCIEBAR	INCR Burst Read
MemWr 32 (3DW) to PCIEBAR	INCR Burst Write
MemRd 64 (4DW) of PCIEBAR	INCR Burst Read
MemWr 64 (4DW) to PCIEBAR	INCR Burst Write

Note: For PCIe® requests with lengths greater than 1 Dword, the size of the data burst on the Master AXI interface will always equal the width of the AXI data bus even when the request received from the PCIe link is shorter than the AXI bus width.

Note: `s_axi_wstrb` can be used to facilitate data alignment to an address boundary. `s_axi_wstrb` may equal 0 in the beginning of a valid data cycle and will appropriately calculate an offset to the given address. However, the valid data identified by `s_axi_wstrb` must be continuous from the first byte enable to the last byte enable.

Transaction Ordering for PCIe

The AXI Bridge for PCI Express Gen3 core conforms to strict PCIe transaction ordering rules. See the PCIe v2.1 Specification [Ref 10] for the complete rule set. The following behaviors are implemented in the AXI Bridge for PCI Express Gen3 core to enforce the PCIe transaction ordering rules on the highly-parallel AXI bus of the bridge. The rules are enforced without regard to the Relaxed Ordering attribute bit within the TLP header:

- The `bresp` to the remote (requesting) AXI4 master device for a write to a remote PCIe device is not issued until the MemWr TLP transmission is guaranteed to be sent on the PCIe link before any subsequent TX-transfers.

- A remote PCIe device read of a remote AXI slave is not permitted to pass any previous remote PCIe device writes to a remote AXI slave received by the AXI Bridge for PCI Express Gen3 core. The AXI read address phase is held until the previous AXI write transactions have completed and `bresp` has been received for the AXI write transactions.
- Read completion data received from a remote PCIe device are not permitted to pass any remote PCIe device writes to a remote AXI slave received by the AXI Bridge for PCI Express Gen3 core prior to the read completion data. The `bresp` for the AXI write(s) must be received before the completion data is presented on the AXI read data channel.

IMPORTANT: *The transaction ordering rules for PCIe might have an impact on data throughput in heavy bidirectional traffic.*

BAR and Address Translation

BAR Addressing

C_AXIBAR_n and C_AXIBAR_HIGHADDR_n are used to calculate the size of the AXI BAR *n* and during address translation to PCIe address.

- C_AXIBAR_n provides the low address where AXI BAR *n* starts and will be regarded as address offset 0x0 when the address is translated.
- C_AXIBAR_HIGHADDR_n is the high address of the last valid byte address of AXI BAR n. (For more details on how the address gets translated, see Address Translation.)

The difference between the two parameters is your AXI BAR *n* size. These parameters must be set accordingly such that the AXI BAR *n* size is a power of two and must have at least 4K.

When a packet is sent to the core (outgoing PCIe packets), the packet must have an address that is in the range of C_AXIBAR_n and C_AXIBAR_HIGHADDR_n. Any packet that is received by the core that has an address outside of this range will be responded to with a SLVERR. When the IP integrator is used, these parameters are derived from the Address Editor tab within the IP integrator. The Address Editor sets the AXI Interconnect as well as the core so the address range matches, and the packet is routed to the core only when the packet has an address within the valid range.

Address Translation

The address space for PCIe is different than the AXI address space. To access one address space from another address space requires an address translation process. On the AXI side, the bridge supports mapping to PCIe on up to six 32-bit or 64-bit AXI base address registers (BARs). The generics used to configure the BARs follow.

C_AXIBAR_NUM, C_AXIBAR_n, C_AXIBAR_HIGHADDR_n, and C_AXIBAR2PCIEBAR_n

where *n* represents an AXIBAR number from 0 to 5. The bridge for PCIe supports mapping on up to six 32-bit BARs or three 64-bit BARs for PCIe. The generics used to configure the BARs are:

PCIEBAR_NUM, C_PCIE2AXIBAR_n and PF0_BARn_APERTURE_SIZE

where *n* represents a particular BAR number for PCIe from 0 to 5.

Note: AXIBAR2PCIEBAR_n translation vectors can be changed by using software by writing to AXI Base Address Translation Configuration Registers.

Four examples follow:

- Example 1 (32-bit PCIe Address Mapping) demonstrates how to set up three AXI BARs and translate the AXI address to a 32-bit address for PCIe.
- Example 2 (64-bit PCIe Address Mapping) demonstrates how to set up three AXI BARs and translate the AXI address to a 64-bit address for PCIe.
- Example 3 demonstrates how to set up two 64-bit PCIe BARs and translate the address for PCIe to an AXI address.
- Example 4 demonstrates how to set up a combination of two 32-bit AXI BARs and two 64 bit AXI BARs, and translate the AXI address to an address for PCIe.

Example 1 (32-bit PCIe Address Mapping)

This example shows the generic settings to set up three independent AXI BARs and address translation of AXI addresses to a remote 32-bit address space for PCIe. This setting of AXI BARs does not depend on the BARs for PCIe within the AXI Bridge for PCI Express Gen3 core.

In this example, where C_AXIBAR_NUM=3, the following assignments for each range are made:

```
AXI_ADDR_WIDTH=48

C_AXIBAR_0=0x00000000_12340000
C_AXI_HIGHADDR_0=0x00000000_1234FFFF (64 Kbytes)
C_AXIBAR2PCIEBAR_0=0x00000000_56710000 (Bits 63-32 are zero in order to produce a
32-bit PCIe TLP. Bits 15-0 must be zero based on the AXI BAR aperture size. Non-zero
values in the lower 16 bits are invalid translation values.)

C_AXIBAR_1=0x00000000_ABCDE000
```

```
C_AXI_HIGHADDR_1=0x00000000_ABCDFFFF (8 Kbytes)
C_AXIBAR2PCIEBAR_1=0x00000000_FEDC0000 (Bits 63-32 are zero in order to produce a
32-bit PCIe TLP. Bits 12-0 must be zero based on the AXI BAR aperture size. Non-zero
values in the lower 13 bits are invalid translation values.)

C_AXIBAR_2=0x00000000_FE000000
C_AXI_HIGHADDR_2=0x00000000_FFFFFFFF (32 Mbytes)
C_AXIBAR2PCIEBAR_2=0x00000000_40000000 (Bits 63-32 are zero in order to produce a
32-bit PCIe TLP. Bits 24-0 must be zero based on the AXI BAR aperture size. Non-zero
values in the lower 25 bits are invalid translation values.)
```

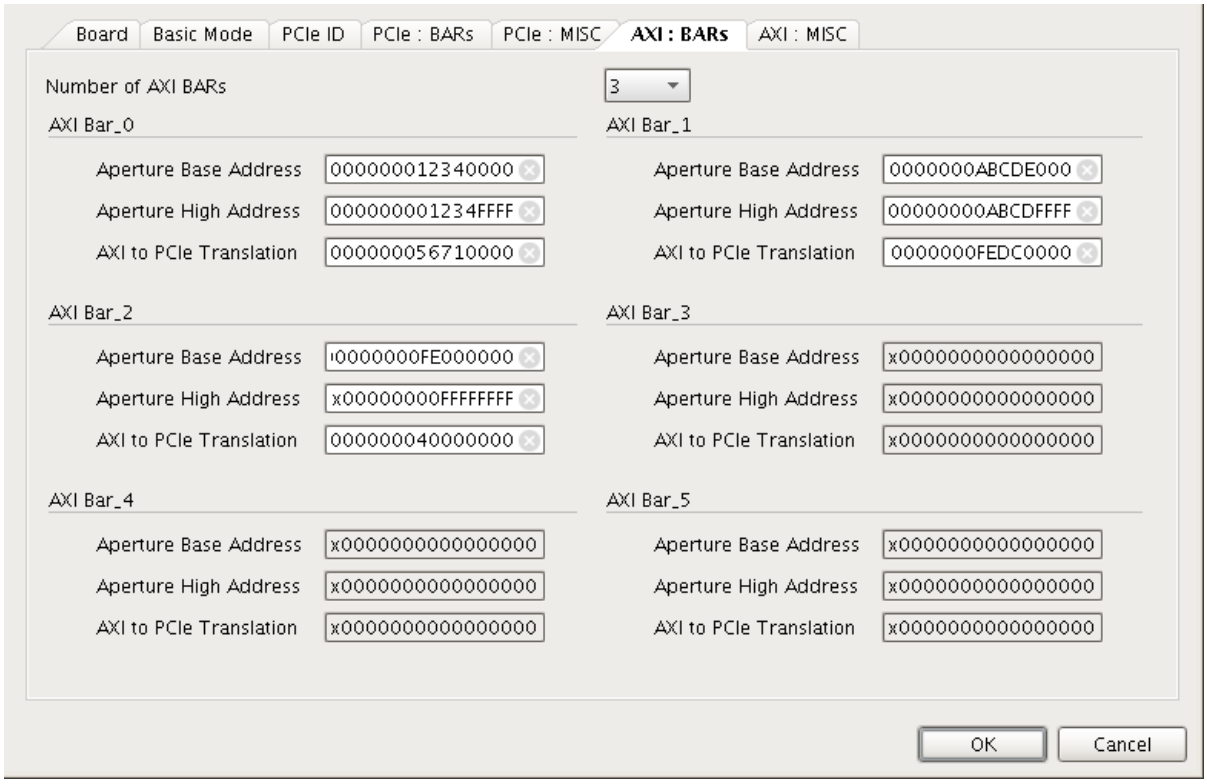

Figure 3-3: **Example 1 Settings**

- Accessing the Bridge AXIBAR_0 with address `0x0000_12340ABC` on the AXI bus yields `0x56710ABC` on the bus for PCIe.

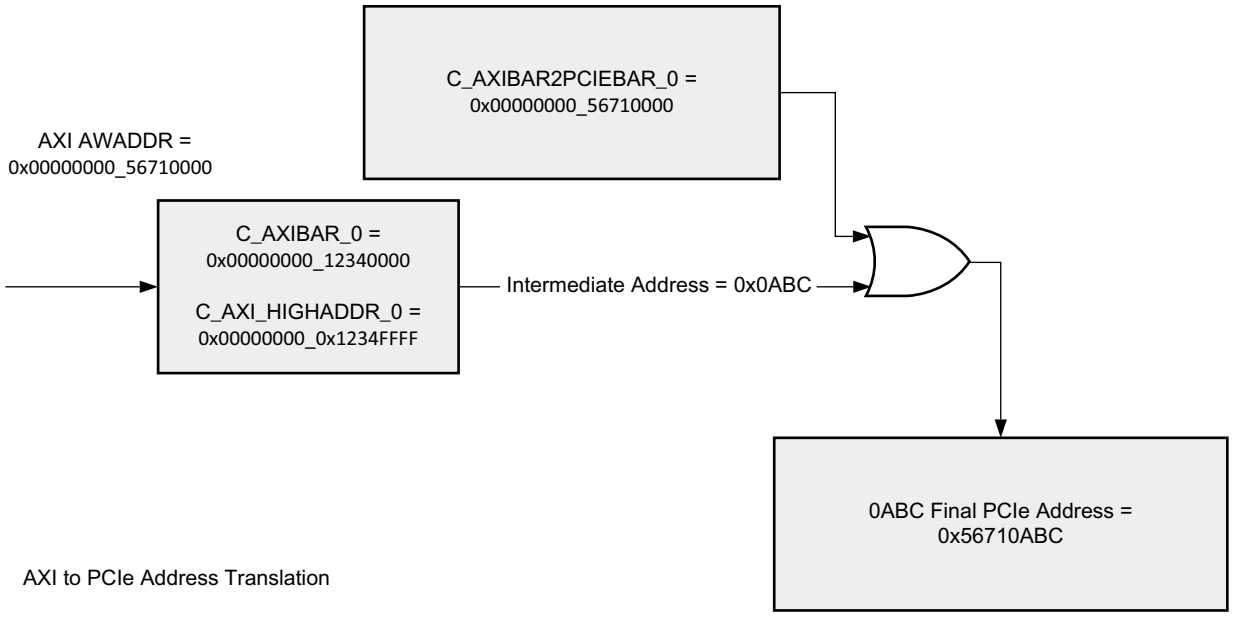

Figure 3-4: **AXI to PCIe Address Translation**

- Accessing the Bridge AXIBAR_1 with address `0x0000_ABCDF123` on the AXI bus yields `0xFEDC1123` on the bus for PCIe.
- Accessing the Bridge AXIBAR_2 with address `0x0000_FFEDCBA` on the AXI bus yields `0x41FEDCBA` on the bus for PCIe.

Example 2 (64-bit PCIe Address Mapping)

This example shows the generic settings to set up to three independent AXI BARs and address translation of AXI addresses to a remote 64-bit address space for PCIe. This setting of AXI BARs does not depend on the BARs for PCIe within the Bridge.

In this example, where C_AXIBAR_NUM=3, the following assignments for each range are made:

```
AXI_ADDR_WIDTH=48

C_AXIBAR_0=0x00000000_1234FFFF
C_AXI_HIGHADDR_0=0x00000000_12340000 (64 Kbytes)
C_AXIBAR2PCIEBAR_0=0x5000000056710000 (Bits 63-32 are non-zero in order to produce a
64-bit PCIe TLP. Bits 15-0 must be zero based on the AXI BAR aperture size. Non-zero
values in the lower 16 bits are invalid translation values.)

C_AXIBAR_1=0x00000000_ABCDE000
C_AXI_HIGHADDR_1=0x00000000_ABCDFFFF (8 Kbytes)
C_AXIBAR2PCIEBAR_1=0x60000000_FEDC0000 (Bits 63-32 are non-zero in order to produce
a 64-bit PCIe TLP. Bits 12-0 must be zero based on the AXI BAR aperture size. Non-zero
values in the lower 13 bits are invalid translation values.)

C_AXIBAR_2=0x00000000_FE000000
```

```
C_AXI_HIGHADDR_2=0x00000000_FFFFFFFF (32 Mbytes)
C_AXIBAR2PCIEBAR_2=0x7000000040000 (Bits 63-32 are non-zero in order to produce a
64-bit PCIe TLP. Bits 24-0 must be zero based on the AXI BAR aperture size. Non-zero
values in the lower 25 bits are invalid translation values.)
```

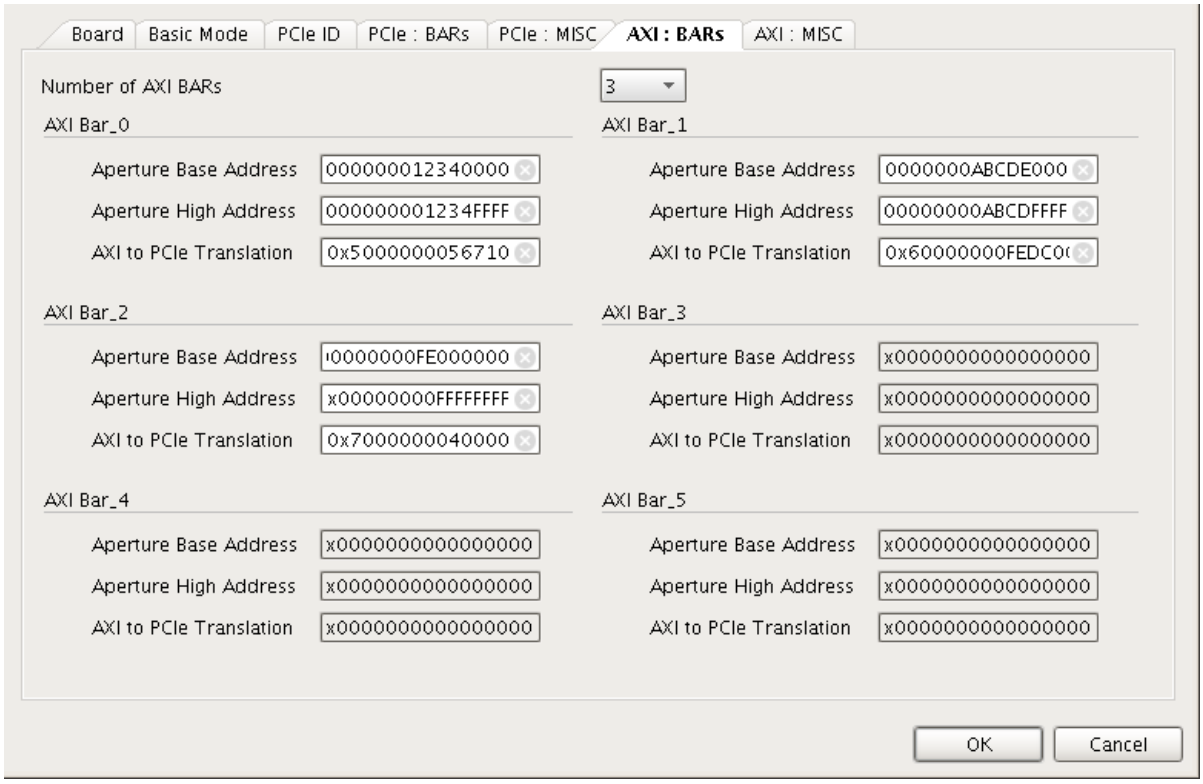

Figure 3-5: **Example 2 Settings**

- Accessing the Bridge AXIBAR_0 with address `0x0000_12340ABC` on the AXI bus yields `0x5000000056710ABC` on the bus for PCIe.
- Accessing the Bridge AXIBAR_1 with address `0x0000_ABCDF123` on the AXI bus yields `0x60000000FEDC1123` on the bus for PCIe.
- Accessing the Bridge AXIBAR_2 with address `0x0000_FFFEDCBA` on the AXI bus yields `0x7000000041FEDCBA` on the bus for PCIe.

Example 3

This example shows the generic settings to set up two independent BARs for PCIe and address translation of addresses for PCIe to a remote AXI address space. This setting of BARs for PCIe does not depend on the AXI BARs within the bridge.

In this example, where C_PCIEBAR_NUM=2, the following range assignments are made:

```
AXI_ADDR_WIDTH=48
```

BAR 0 is set to 0x20000000_ABCD8000 by the Root Port. (Since this is a 64-bit BAR PCIe, BAR1 is disabled.)
PF0_BAR0_APERTURE_SIZE=0x08 (32 Kbytes)
C_PCIEBAR2AXIBAR_0=0x00000000_12340000 (Because the AXI address is 48-bits wide, bits 63-48 should be zero. Base on the PCIe Bar Size bits 14-0 should be zero. Non-zero values in these ranges are invalid.)

BAR 2 is set to 0xA000000012000000 by Root Port. (Since this is a 64-bit BAR PCIe BAR3 is disabled.)
PF0_BAR0_APERTURE_SIZE=0x12 (32 Mbytes)
C_PCIEBAR2AXIBAR_2=0x00000000_FE000000 (Because the AXI address is 48-bits wide, bits 63-48 should be zero. Base on the PCIe Bar Size bits 24-0 should be zero. Non-zero values in these ranges are invalid.)

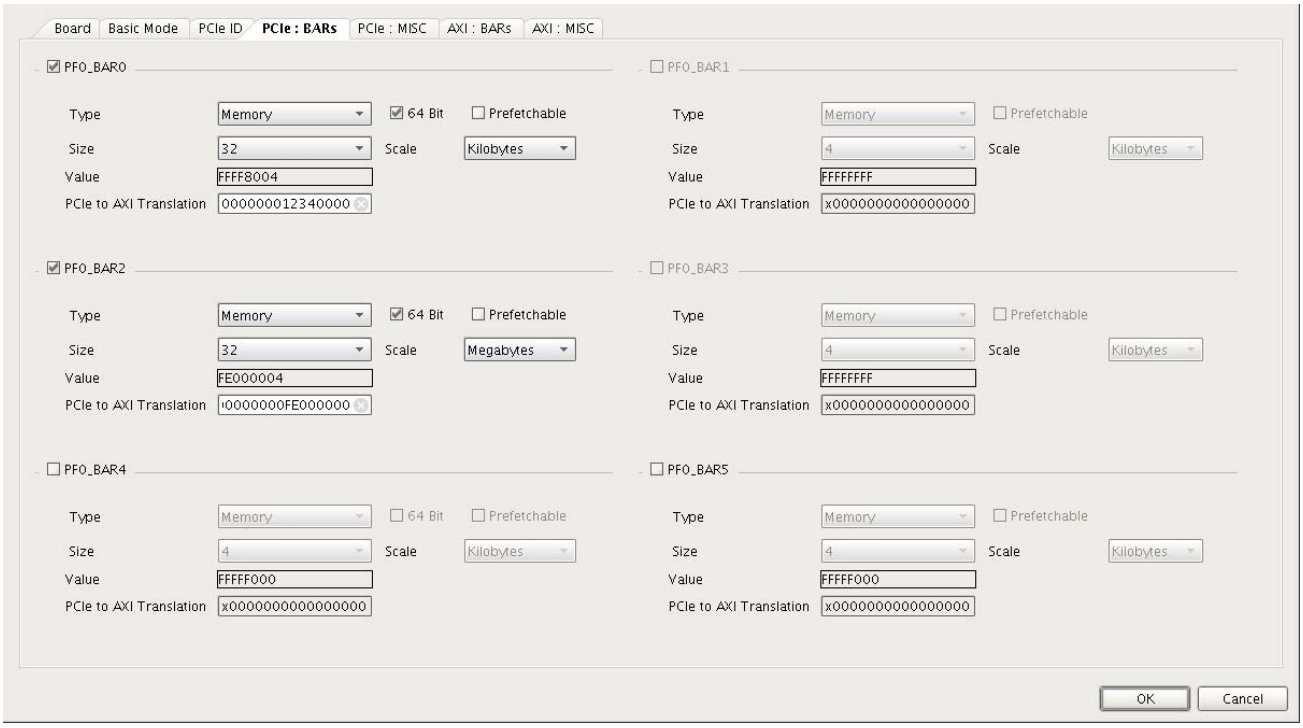

Figure 3-6: **Example 3 Settings**

- Accessing the Bridge PCIEBAR_0 with address 0x20000000_ABCDFFF4 on the bus for PCIe yields 0x0000_12347FF4 on the AXI bus.

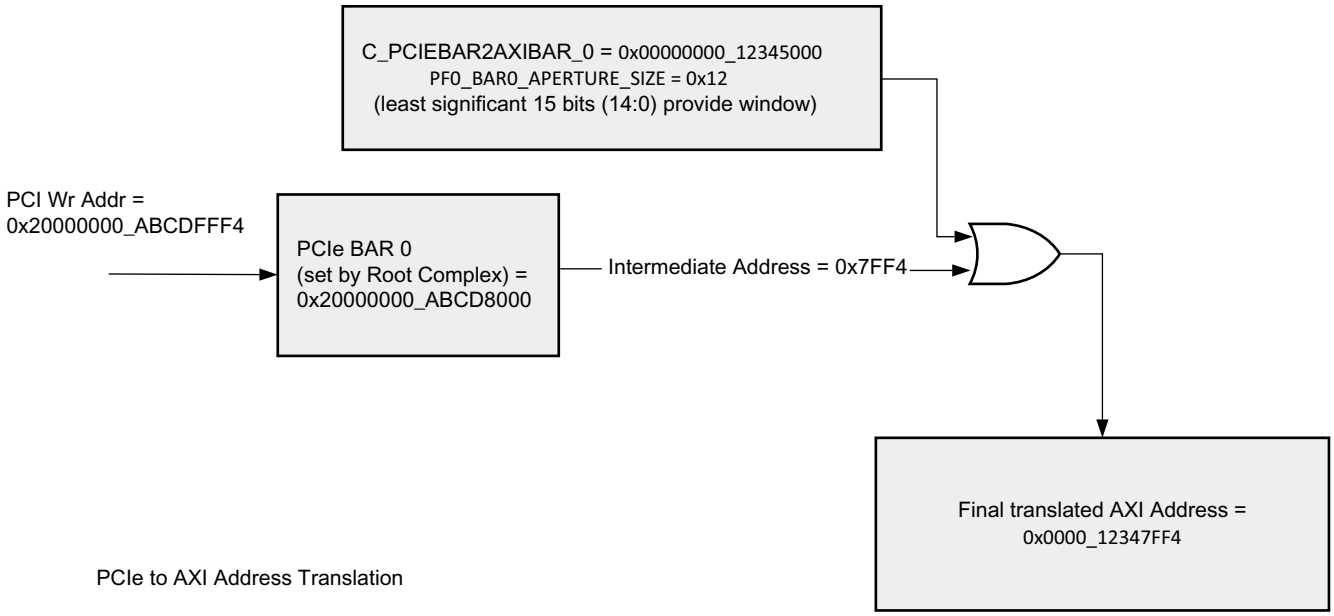

Figure 3-7: **PCIe to AXI Translation**

- Accessing Bridge PCIEBAR_2 with address `0xA00000001235FEDC` on the bus for PCIe yields `0x0000_FE35FEDC` on the AXI bus.

Example 4

This example shows the generic settings of four AXI BARs and address translation of AXI addresses to a remote 32-bit and 64-bit addresses for PCIe. This setting of AXI BARs do not depend on the BARs for PCIe within the Bridge.

In this example, where C_AXIBAR_NUM=4, the following assignments for each range are made:

```
AXI_ADDR_WIDTH=48

C_AXIBAR_0=0x00000000_12340000
C_AXI_HIGHADDR_0=0x00000000_1234FFFF (64 KB)
C_AXIBAR2PCIEBAR_0=0x00000000_56710000 (Bits 63-32 are zero to produce a 32-bit PCIe
TLP. Bits 15-0 must be zero based on the AXI BAR aperture size. Non-zero values in
the lower 16 bits are invalid translation values.)

C_AXIBAR_1=0x00000000_ABCDE000
C_AXI_HIGHADDR_1=0x00000000_ABCDFFFF (8 KB)
C_AXIBAR2PCIEBAR_1=0x50000000_FEDC0000 (Bits 63-32 are non-zero to produce a 64-bit
PCIe TLP. Bits 12-0 must be zero based on the AXI BAR aperture size. Non-zero values
in the lower 13 bits are invalid translation values.)

C_AXIBAR_2=0x00000000_FE000000
C_AXI_HIGHADDR_2=0x00000000_FFFFFFFF (32 MB)
C_AXIBAR2PCIEBAR_2=0x00000000_40000000 (Bits 63-32 are zero to produce a 32-bit PCIe
TLP. Bits 24-0 must be zero based on the AXI BAR aperture size. Non-zero values in
the lower 25 bits are invalid translation values.)
```

```
C_AXIBAR_3=0x00000000_00000000
C_AXI_HIGHADDR_3=0x00000000_00000FFF (4 KB)
C_AXIBAR2PCIEBAR_3=0x60000000_87654000 (Bits 63-32 are non-zero to produce a 64-bit
PCIe TLP. Bits 11-0 must be zero based on the AXI BAR aperture size. Non-zero values
in the lower 12 bits are invalid translation values.)
```

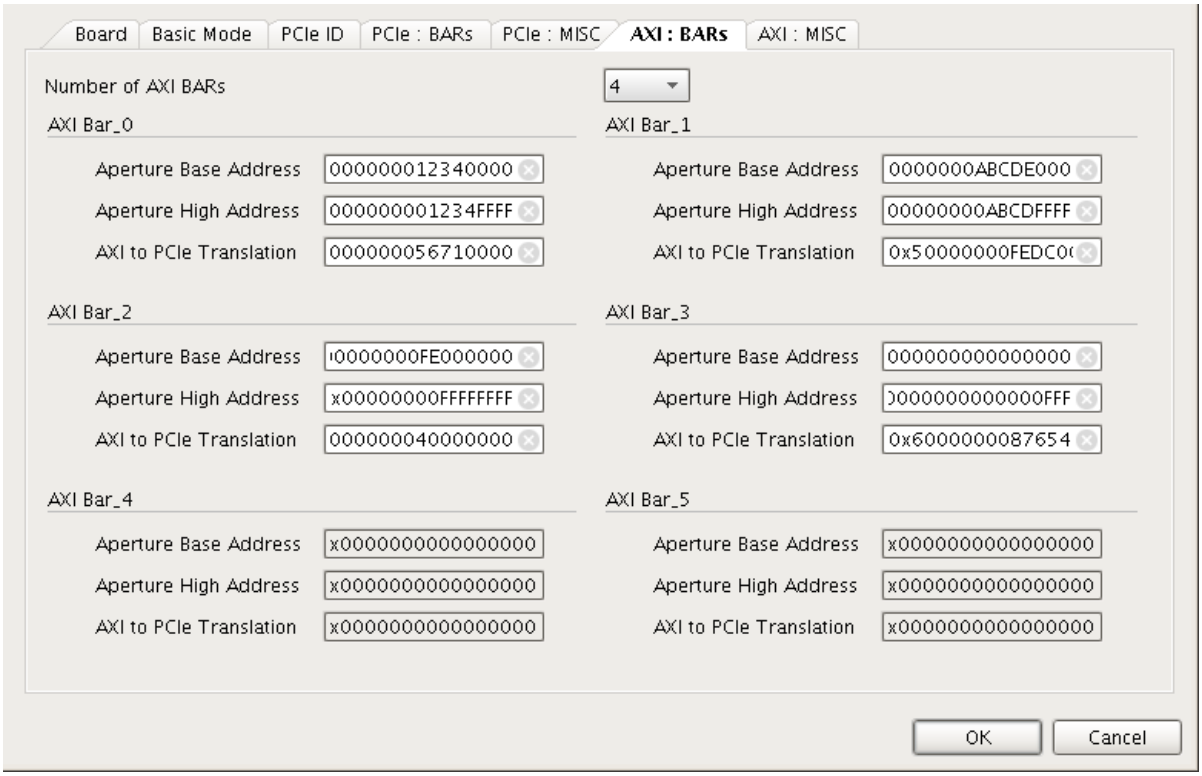

Figure 3-8: **Example 4 Settings**

- Accessing the Bridge AXIBAR_0 with address `0x0000_12340ABC` on the AXI bus yields `0x56710ABC` on the bus for PCIe.
- Accessing the Bridge AXIBAR_1 with address `0x0000_ABCDF123` on the AXI bus yields `0x50000000FEDC1123` on the bus for PCIe.
- Accessing the Bridge AXIBAR_2 with address `0x0000_FFFEDCBA` on the AXI bus yields `0x41FEDCBA` on the bus for PCIe.
- Accessing the AXI M S PCIe Bridge AXIBAR_3 with address `0x0000_00000071` on the AXI bus yields `0x6000000087654071` on the bus for PCIe.

Addressing Checks

When setting the following parameters for PCIe address mapping, C_PCIE2AXIBAR_n and PF0_BARn_APERTURE_SIZE, be sure these are set to allow for the addressing space on the AXI system. For example, the following setting is illegal and results in an invalid AXI address.

```
C_PCIE2AXIBAR_0 = 0x00000000_FFFFF000
```

```
PF0_BARn_APERTURE_SIZE=0x06 (8 KB)
```

For an 8 Kilobyte BAR the lower 13 bits must be zero. As result, the C_PCIE2AXIBAR_0 value should be modified to be `0x00000000_FFFFE000`. Also, check for a larger value on PF0_BARn_APERTURE_SIZE compared to the value assigned to parameter, C_PCIE2AXIBAR_n. For example, the following parameter settings.

```
C_PCIE2AXIBAR_0 = 0xFFFF_E000
PF0_BARn_APERTURE_SIZE=0x0D (1 MB)
```

To keep the AXIBAR upper address bits as 0xFFFF_E000 (to reference bits [31:13]), the PF0_BARn_APERTURE_SIZE parameter must be set to 0x06 (8 KB).

Interrupts

Interrupt capabilities are provided by the underlying PCI Express solution IP. For additional information, see the *Virtex-7 FPGAs Gen3 Integrated Block for PCI Express Product Guide* (PG023) [Ref 4] and the *UltraScale Architecture Gen3 Integrated Block for PCI Express Product Guide* (PG156) [Ref 5].

Malformed TLP

The integrated block for PCI Express detects a malformed TLP. For the IP configured as an Endpoint core, a malformed TLP results in a fatal error message being sent upstream if error reporting is enabled in the Device Control register.

Abnormal Conditions

This section describes how the Slave side (Table 3-3) and Master side (Table 3-4) of the AXI Bridge for PCI Express Gen3 core handle abnormal conditions.

Slave Bridge Abnormal Conditions

Slave bridge abnormal conditions are classified as: Illegal Burst Type and Completion TLP Errors. The following sections describe the manner in which the Bridge handles these errors.

Illegal Burst Type

The slave bridge monitors AXI read and write burst type inputs to ensure that only the INCR (incrementing burst) type is requested. Any other value on these inputs is treated as an error condition and the Slave Illegal Burst (SIB) interrupt is asserted. In the case of a read request, the Bridge asserts SLVERR for all data beats and arbitrary data is placed on the `s_axi_rdata` bus. In the case of a write request, the Bridge asserts SLVERR for the write response and all write data is discarded.

Completion TLP Errors

Any request to the bus for PCIe (except for a posted Memory write) requires a completion TLP to complete the associated AXI request. The Slave side of the Bridge checks the received completion TLPs for errors and checks for completion TLPs that are never returned (Completion Timeout). Each of the completion TLP error types are discussed in the subsequent sections.

Unexpected Completion

When the slave bridge receives a completion TLP, it matches the header RequesterID and Tag to the outstanding RequesterID and Tag. A match failure indicates the TLP is an Unexpected Completion which results in the completion TLP being discarded and a Slave Unexpected Completion (SUC) interrupt strobe being asserted. Normal operation then continues.

Unsupported Request

A device for PCIe might not be capable of satisfying a specific read request. For example, the read request targets an unsupported address for PCIe causing the completer to return a completion TLP with a completion status of 0b001 - Unsupported Request. The completer can also return a completion TLP with a completion status that is reserved according to the 2.1 PCIe Specification, which must be treated as an unsupported request status. When the slave bridge receives an unsupported request response, the Slave Unsupported Request (SUR) interrupt is asserted and the DECERR response is asserted with arbitrary data on the memory mapped AXI4 bus.

Completion Timeout

A Completion Timeout occurs when a completion (Cpl) or completion with data (CplD) TLP is not returned after an AXI to PCIe read request. Completions must complete within the C_COMP_TIMEOUT parameter selected value from the time the MemRd for PCIe request is issued. When a completion timeout occurs, a Slave Completion Timeout (SCT) interrupt is asserted and the SLVERR response is asserted with arbitrary data on the memory mapped AXI4 bus.

Poison Bit Received on Completion Packet

An Error Poison occurs when the completion TLP EP bit is set, indicating that there is poisoned data in the payload. When the slave bridge detects the poisoned packet, the Slave Error Poison (SEP) interrupt is asserted and the SLVERR response is asserted with arbitrary data on the memory mapped AXI4 bus.

Completer Abort

A Completer Abort occurs when the completion TLP completion status is 0b100 - Completer Abort. This indicates that the completer has encountered a state in which it was unable to complete the transaction. When the slave bridge receives a completer abort response, the Slave Completer Abort (SCA) interrupt is asserted and the SLVERR response is asserted with arbitrary data on the memory mapped AXI4 bus.

Table 3-3: **Slave Bridge Response to Abnormal Conditions**

Transfer Type	Abnormal Condition	Bridge Response
Read	Illegal burst type	SIB interrupt is asserted. SLVERR response given with arbitrary read data.
Write	Illegal burst type	SIB interrupt is asserted. Write data is discarded. SLVERR response given.
Read	Unexpected completion	SUC interrupt is asserted. Completion is discarded.
Read	Unsupported Request status returned	SUR interrupt is asserted. DECERR response given with arbitrary read data.
Read	Completion timeout	SCT interrupt is asserted. SLVERR response given with arbitrary read data.
Read	Poison bit in completion	Completion data is discarded. SEP interrupt is asserted. SLVERR response given with arbitrary read data.
Read	Completer Abort (CA) status returned	SCA interrupt is asserted. SLVERR response given with arbitrary read data.

Master Bridge Abnormal Conditions

The following sections describe the manner in which the master bridge handles abnormal conditions.

AXI DECERR Response

When the master bridge receives a DECERR response from the AXI bus, the request is discarded and the Master DECERR (MDE) interrupt is asserted. If the request was

non-posted, a completion packet with the Completion Status = Unsupported Request (UR) is returned on the bus for PCIe.

AXI SLVERR Response

When the master bridge receives a SLVERR response from the addressed AXI slave, the request is discarded and the Master SLVERR (MSE) interrupt is asserted. If the request was non-posted, a completion packet with the Completion Status = Completer Abort (CA) is returned on the bus for PCIe.

Max Payload Size for PCIe, Max Read Request Size or 4K Page Violated

It is the responsibility of the requester to ensure that the outbound request adheres to the Max Payload Size, Max Read Request Size, and 4 Kb Page Violation rules. If the master bridge receives a request that violates one of these rules, the bridge processes the invalid request as a valid request, which can return a completion that violates one of these conditions or can result in the loss of data. The master bridge does not return a malformed TLP completion to signal this violation.

Completion Packets

When the MAX_READ_REQUEST_SIZE is greater than the MAX_PAYLOAD_SIZE, a read request for PCIe can ask for more data than the master bridge can insert into a single completion packet. When this situation occurs, multiple completion packets are generated up to MAX_PAYLOAD_SIZE, with the Read Completion Boundary (RCB) observed.

Poison Bit

When the poison bit is set in a transaction layer packet (TLP) header, the payload following the header is corrupted. When the master bridge receives a memory request TLP with the poison bit set, it discards the TLP and asserts the Master Error Poison (MEP) interrupt strobe.

Zero Length Requests

When the master bridge receives a read request with the Length = 0x1, FirstBE = 0x00, and LastBE = 0x00, it responds by sending a completion with Status = Successful Completion. When the master bridge receives a write request with the Length = 0x1, FirstBE = 0x00, and LastBE = 0x00 there is no effect.

Table 3-4: **Master Bridge Response to Abnormal Conditions**

Transfer Type	Abnormal Condition	Bridge Response
Read	DECERR response	MDE interrupt strobe asserted. Completion returned with Unsupported Request status.
Write	DECERR response	MDE interrupt strobe asserted.

Table 3-4: **Master Bridge Response to Abnormal Conditions**

Transfer Type	Abnormal Condition	Bridge Response
Read	SLVERR response	MSE interrupt strobe asserted. Completion returned with Completer Abort status.
Write	SLVERR response	MSE interrupt strobe asserted.
Write	Poison bit set in request	MEP interrupt strobe asserted. Data is discarded.
Read	DECERR response	MDE interrupt strobe asserted. Completion returned with Unsupported Request status.
Write	DECERR response	MDE interrupt strobe asserted.

Link Down Behavior

The normal operation of the AXI Bridge for PCI Express Gen3 core is dependent on the integrated block for PCIe establishing and maintaining the point-to-point link with an external device for PCIe. If the link has been lost, it must be re-established to return to normal operation.

When a Hot Reset is received by the AXI Bridge for PCI Express Gen3 core, the link goes down and the PCI Configuration Space must be reconfigured.

Initiated AXI4 write transactions that have not yet completed on the AXI4 bus when the link goes down have a SLVERR response given and the write data is discarded. Initiated AXI4 read transactions that have not yet completed on the AXI4 bus when the link goes down have a SLVERR response given, with arbitrary read data returned.

Any MemWr TLPs for PCIe that have been received, but the associated AXI4 write transaction has not started when the link goes down, are discarded. If the associated AXI4 write transaction is in the process of being transferred, it completes as normal. Any MemRd TLPs for PCIe that have been received, but have not returned completion TLPs by the time the link goes down, complete on the AXI4 bus, but do not return completion TLPs on the PCIe bus.

Root Port

When configured to support Root Port functionality, the AXI Bridge for PCI Express Gen3 core fully supports Root Port operation as supported by the underlying block. There are a few details that need special consideration. The following subsections contain information and design considerations about Root Port support.

Power Limit Message TLP

The AXI Bridge for PCI Express Gen3 core automatically sends a Power Limit Message TLP when the Master Enable bit of the Command Register is set. The software must set the Requester ID register before setting the Master Enable bit to ensure that the desired Requester ID is used in the Message TLP.

Root Port Configuration Read

When an ECAM access is performed to the primary bus number, self-configuration of the integrated block for PCIe is performed. A PCIe configuration transaction is not performed and is not presented on the link. When an ECAM access is performed to the bus number that is equal to the secondary bus value in the Enhanced PCIe type 1 configuration header, then type 0 configuration transactions are generated.

When an ECAM access is attempted to a bus number that is in the range defined by the secondary bus number and subordinate bus number range (not including secondary bus number), then type 1 configuration transactions are generated. The primary, secondary and subordinate bus numbers are written and updated by Root Port software to the type 1 PCI™ Configuration Header of the AXI Bridge for PCI Express Gen3 core in the enumeration procedure.

When an ECAM access is attempted to a bus number that is out of the range defined by the secondary bus_number and subordinate bus number, the bridge does not generate a configuration request and signal a SLVERR response on the AXI4-Lite bus.

When a Unsupported Request (UR) response is received for a configuration read request, all ones are returned on the AXI4-Lite bus to signify that a device does not exist at the requested device address. It is the responsibility of the software to ensure configuration write requests are not performed to device addresses that do not exist. However, the AXI Bridge for PCI Express Gen3 core asserts SLVERR response on the AXI4-Lite bus when a configuration write request is performed on device addresses that do not exist or a UR response is received.

Root Port BAR

During core customization in the Vivado Design Suite, when there is no BAR enabled, RP passes all received packets to the user application without address translation or address filtering.

When BAR is enabled, by default the BAR address starts at 0x0000_0000 unless programmed separately. Any packet received from the PCIe link that hits a BAR is translated according to the PCIE-to-AXI Address Translation rules. All other packets that do not hit a BAR are passed to the user without address translation or address filtering.

The Root Port BAR customization options in the Vivado Design Suite are found in PCIe Base Address Registers in Chapter 4.

Configuration Transaction Timeout

Configuration transactions are non-posted transactions. The AXI Bridge for PCI Express Gen3 core has a timer for timeout termination of configuration transactions that have not completed on the PCIe link. `SLVERR` is returned when a configuration timeout occurs. Timeouts of configuration transactions are flagged by an interrupt as well.

Abnormal Configuration Transaction Termination Responses

Responses on AXI4-Lite to abnormal terminations to configuration transactions are shown in Table 3-5.

Table 3-5: **Responses of AXI Bridge for PCI Express Gen3 to Abnormal Configuration Terminations**

Transfer Type	Abnormal Condition	Bridge Response
Config Read or Write	Bus number not in the range of primary bus number through subordinate bus number.	SLVERR response is asserted.
Config Read or Write	Valid bus number and completion timeout occurs.	SLVERR response is asserted.
Config Read or Write	Completion timeout.	SLVERR response is asserted.
Config Write	Bus number in the range of secondary bus number through subordinate bus number and UR is returned.	SLVERR response is asserted.

Tandem Configuration

Tandem Configuration features are available for the AXI Bridge for PCI Express Gen3 core for all UltraScale devices. Tandem Configuration utilizes a two-stage methodology that enables the IP to meet the configuration time requirements indicated in the PCI Express Specification. Multiple use cases are supported with this technology:

- **Tandem PROM**: Load the single two-stage bitstream from the flash.
- **Tandem PCIe**: Load the first stage bitstream from flash, and deliver the second stage bitstream over the PCIe link to the MCAP.
- **Tandem with Field Updates**: After a Tandem PROM or Tandem PCIe initial configuration, update the entire user design while the PCIe link remains active. The update region (floorplan) and design structure are predefined, and Tcl scripts are provided.
- **Tandem + Partial Reconfiguration**: This is a more general case of Tandem Configuration followed by Partial Reconfiguration (PR) of any size or number of PR regions.
- **Partial Reconfiguration over PCIe**: This is a standard configuration followed by PR, using the PCIe / MCAP as the delivery path of partial bitstreams.

To enable any of these capabilities, select the appropriate option when customizing the core. In the Basic tab:

1. Change the **Mode** to **Advanced**.
2. Change the **Tandem Configuration or Partial Reconfiguration** option according to your particular case:
   - **Tandem** for Tandem PROM, Tandem PCIe or Tandem + Partial Reconfiguration use cases.
   - **Tandem with Field Updates** ONLY for the predefined Field Updates use case.
   - **PR over PCIe** to enable the MCAP link for Partial Reconfiguration, without enabling Tandem Configuration.

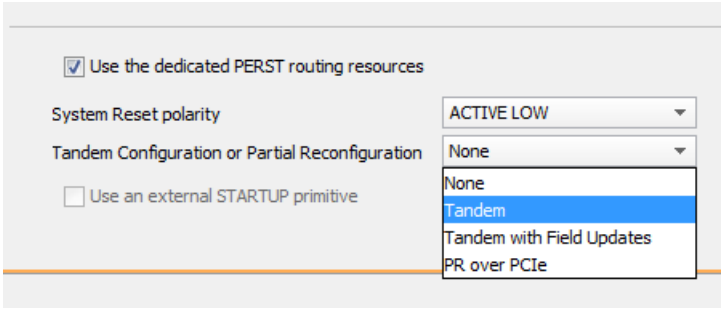

Figure 3-9: **Tandem Configuration or Partial Reconfiguration Option**

For complete information about Tandem Configuration, including required PCIe block locations, design flow examples, requirements, restrictions and other considerations, see Tandem Configuration in the *UltraScale Architecture Gen3 Integrated Block for PCI Express LogiCORE IP Product Guide* (PG156) [Ref 5]. For information on Partial Reconfiguration, see the *Vivado Design Suite User Guide: Partial Reconfiguration* (UG909) [Ref 17].

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 Vivado® design flows and the Vivado 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 18]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 14]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 13]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 15]

Customizing and Generating the Core

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

Note: 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 18] 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 14] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 13].

Note: Figures in this chapter are illustrations of the Vivado Integrated Design Environment (IDE). This layout might vary from the current version.

Customizing the Core

The AXI Bridge for PCI Express Gen3 core customization parameters are described in the following sections.

Basics Parameter Settings

The initial customization screen shown in Figure 4-1 is used to define the basic parameters for the core, including the component name, PCIe® configuration, AXI parameters, and reference clock. Figure 4-2 and Figure 4-3 show the parameters available for the Advanced Mode.

Note: In the Basic Mode tab, the additional parameters available in Advanced Mode are applicable to UltraScale™ architecture devices only.

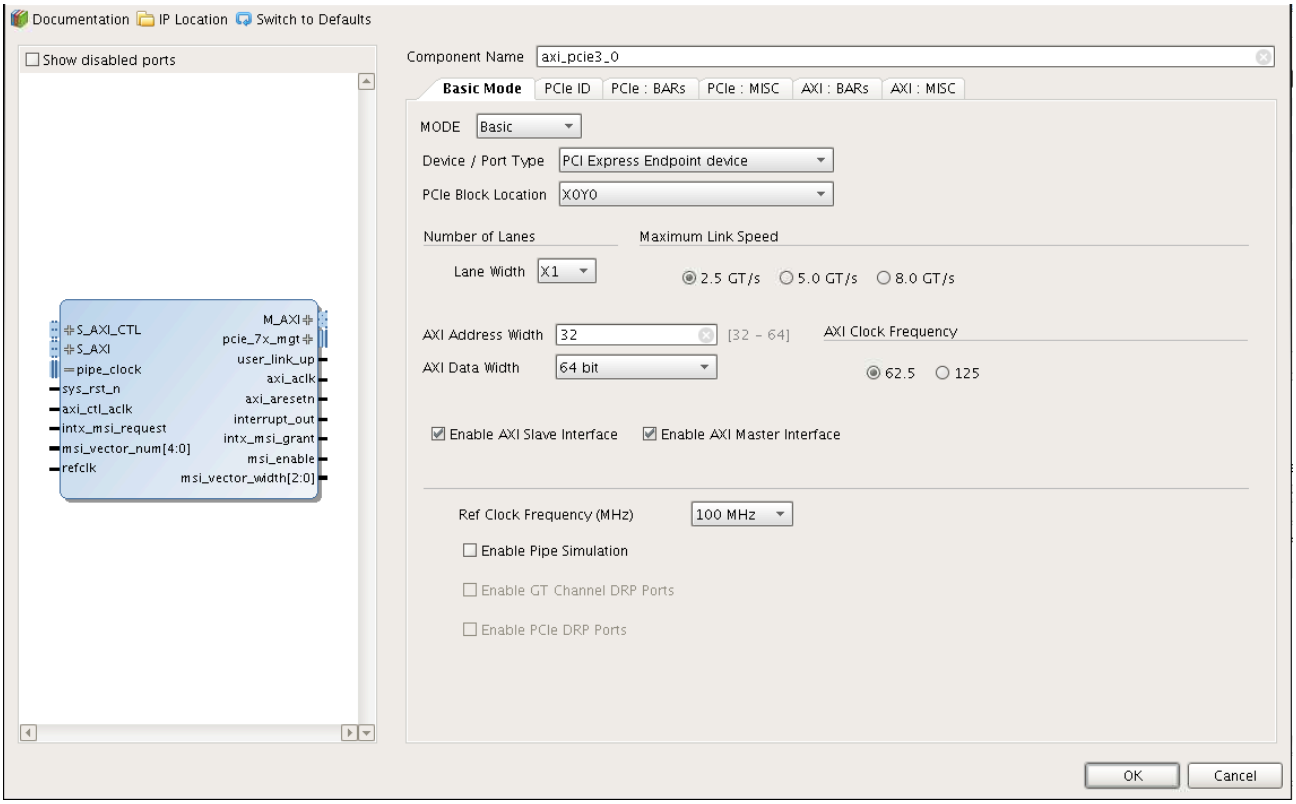

Figure 4-1: **Basics Parameter Settings: Basic Mode Selected**

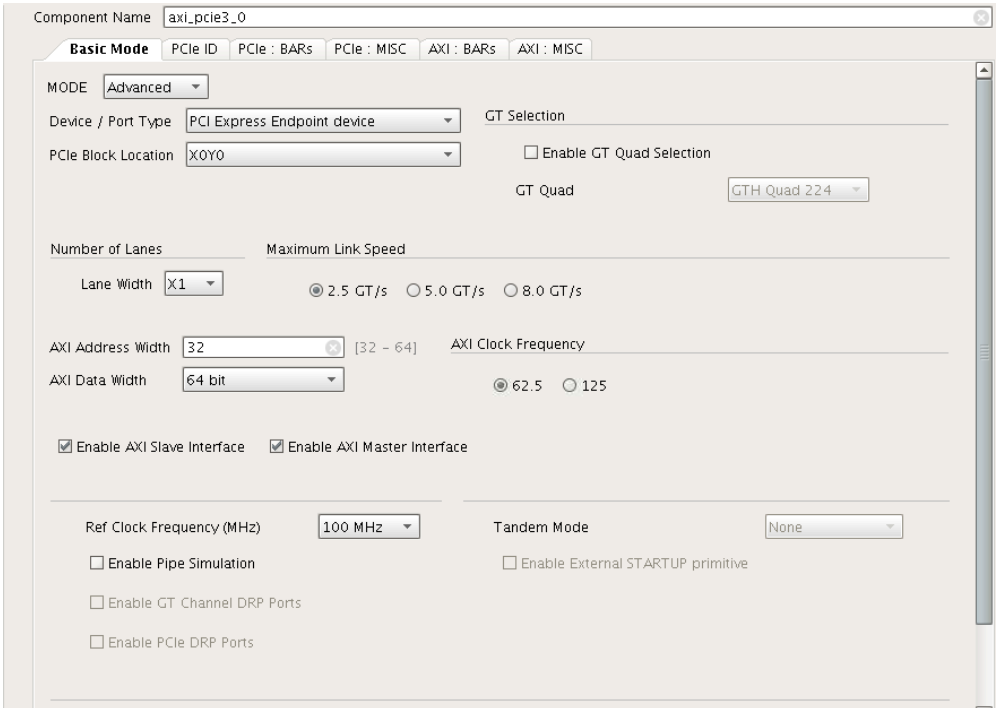

Figure 4-2: **Basic Parameter Settings: Advanced Mode Selected**

Figure 4-3: **Additional Basic Parameter Settings: Advanced Mode Selected**

Component Name

Base name of the output files generated for the core. The name must begin with a letter and can be composed of these characters: a to z, 0 to 9, and "_."

IMPORTANT: *The name cannot be the same as a core module name; for example, "axi_pcie" is a reserved name.*

Mode

Allows you to select the Basic or Advanced mode of the configuration of core.

Device / Port Type

Indicates the PCI Express logical device type.

PCIe Block Location

Selects from the available integrated blocks to enable generation of location-specific constraint files and pinouts. This selection is used in the default example design scripts.

Enable GT Quad Selection

This parameter is used to enable the device/package migration. Applicable to UltraScale devices only.

Number of Lanes

The core requires the selection of the initial lane width. Wider lane width cores can train down to smaller lane widths and consume more FPGA resource.

Maximum Link Speed

Indicates the maximum link speed supported by the design. Higher link speed cores are capable of training to lower link speeds and run at a higher clock frequency.

AXI Address Width

Indicates the AXI address width for the S_AXI and M_AXI interfaces, but does not affect the address width of the S_AXI_CTL interface.

AXI Data Width

Indicates the AXI data width for the S_AXI and M_AXI interfaces, but does not affect the data width of the S_AXI_CTL interface.

AXI Clock Frequency

Indicates the clock frequency that will be generated on the `axi_aclk` output. All AXI interfaces and a majority of the core outputs are synchronous to this clock.

Enable AXI Slave Interface

Allows the slave bridge to be enabled or disabled as desired by the system design. If only the master bridge is used the slave bridge can be disabled to conserve FPGA resources.

Enable AXI Master Interface

Allows the master bridge to be enabled or disabled as desired by the system design. If only the slave bridge is used, the master bridge can be disabled to conserve FPGA resources.

Reference Clock Frequency

Selects the frequency of the reference clock provided on the `refclk` reference clock input. This reference clock input corresponds to `refclk` for Virtex®-7 devices and `sys_clk_gt` for UltraScale devices.

Enable Pipe Simulation

When selected, this option generates the core that can be simulated with PIPE interfaces connected.

Enable GT Channel DRP Ports

When checked, enables the GT channel DRP interface.

Enable PCIe DRP Ports

When checked, enables the PCIe DRP interface.

Tandem Mode

For supported devices only, this option allow you to choose the Tandem Configuration mode: None, Tandem PROM and Tandem PCIe.

Enable External STARTUP primitive

When checked, generates the STARTUP primitive external to the IP.

Use the dedicated PERST routing resources

Enables sys_rst dedicated routing for the PCIE_X0Y0 block.

System Reset polarity

This parameter is used to set the polarity of the `sys_rst` ACTIVE_HIGH or ACTIVE_LOW.

CORE CLOCK Frequency

Available only when an UltraScale device is selected.

This parameter allows you to select the core clock frequencies.

For Gen3 link speed:

- The values of 250 MHz and 500 MHz are available for selection for speed grade -2 or -3 and link width other than x8. For this configuration, this parameter is available when **Advanced** mode is selected.
- For speed grades -2 or -3 and link width of x8, this parameter defaults to 500 MHz and is not available for selection.

- For -1 speed grade (-1, -1L, -1LV,-1H and -1HV) and link width other than x8, this parameter defaults to 250 MHz and is not available for selection.

For Gen1 and Gen2 link speeds:

- This parameter defaults to 250 MHz and is not available for selection.

Note: When -1 or -1L, -1LV, -1H and -1HV speed grade is selected and non production parts of XCKU060 (ES2), XCKU115 (ES2) and VU440 (ES2) are selected, this parameter defaults to 250 MHz and is not available for selection.

PCIe ID Settings

The Identity Settings pages are shown in Figure 4-4. These settings customize the IP initial values and device class code.

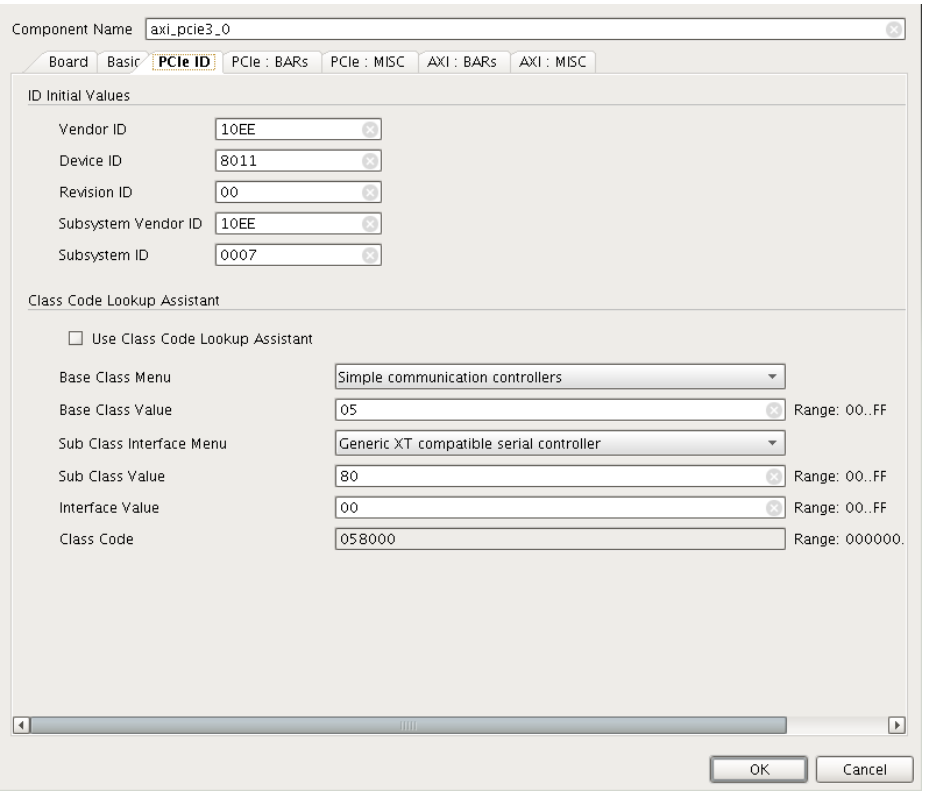

Figure 4-4: **PCIe ID Settings**

ID Initial Values

- **Vendor ID**: Identifies the manufacturer of the device or application. Valid identifiers are assigned by the PCI™ Special Interest Group to guarantee that each identifier is unique. The default value, `10EEh`, is the Vendor ID for Xilinx. Enter your vendor identification number here. `FFFFh` is reserved.

- **Device ID**: A unique identifier for the application; the default value, which depends on the configuration selected, is 70<*link speed*><*link width*>h. This field can be any value; change this value for the application.
- **Revision ID**: Indicates the revision of the device or application; an extension of the Device ID. The default value is `00h`; enter values appropriate for the application.
- **Subsystem Vendor ID**: Further qualifies the manufacturer of the device or application. Enter a Subsystem Vendor ID here; the default value is `10EEh`. Typically, this value is the same as Vendor ID. Setting the value to `0000h` can cause compliance testing issues.
- **Subsystem ID**: Further qualifies the manufacturer of the device or application. This value is typically the same as the Device ID; the default value depends on the lane width and link speed selected. Setting the value to `0000h` can cause compliance testing issues.

Class Code

The Class Code identifies the general function of a device, and is divided into three byte-size fields. The Vivado IDE allows you to either enter the 24-bit value manually (default) by either selecting the **Enter Class Code Manually** checkbox or using the Class Code lookup assistant to populate the field. De-select the checkbox to enable the Class Code assistant.

- **Base Class**: Broadly identifies the type of function performed by the device.
- **Sub-Class**: More specifically identifies the device function.
- **Interface**: Defines a specific register-level programming interface, if any, allowing device-independent software to interface with the device.

Class code encoding can be found in the PCI-SIG® specifications [Ref 10].

Class Code Look-up Assistant

The Class Code Look-up Assistant provides the Base Class, Sub-Class and Interface values for a selected general function of a device. This Look-up Assistant tool only displays the three values for a selected function. You must enter the values in Class Code for these values to be translated into device settings.

PCIe Base Address Registers

The PCIe Base Address Registers (BARs) screen shown in Figure 4-5 sets the base address register space for the Endpoint configuration. Each BAR (0 through 5) configures the BAR Aperture Size and Control attributes of the Physical Function, as described in Table 4-1.

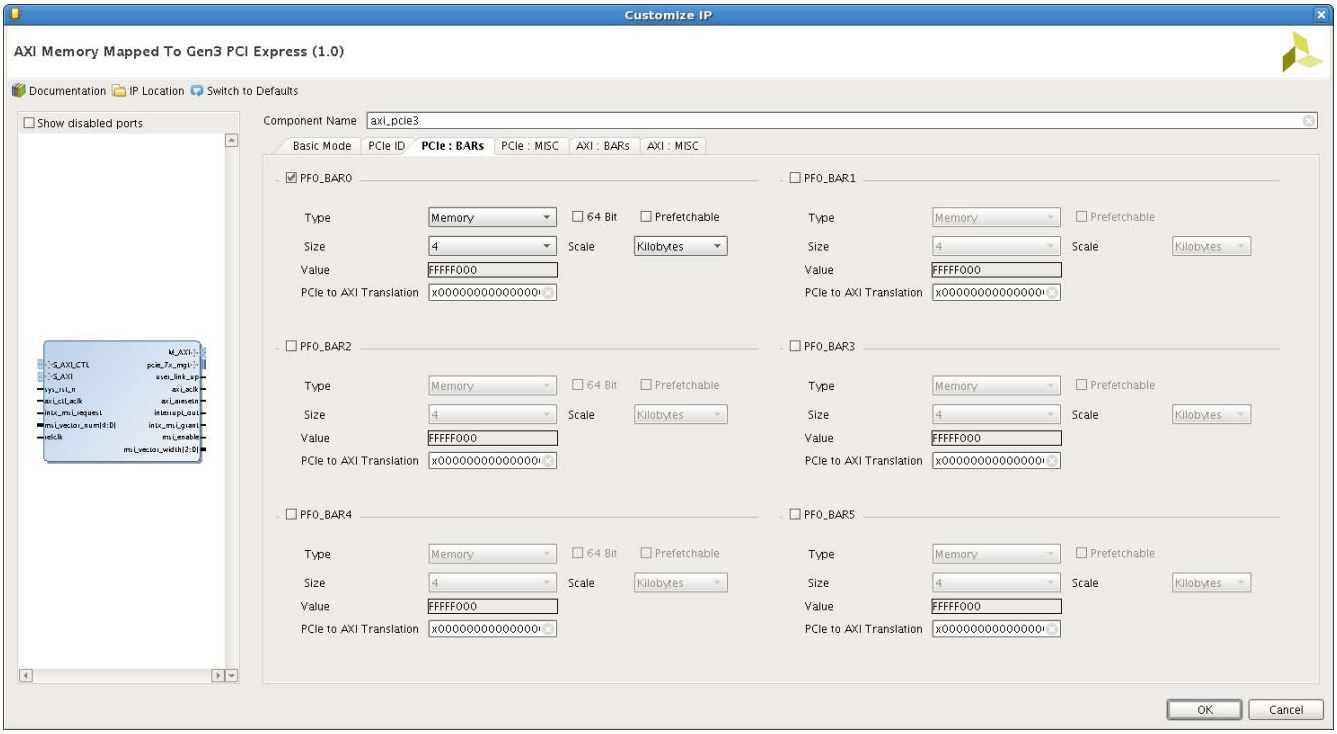

Figure 4-5: **PCIe Base Address Register**

Base Address Register Overview

The AXI Bridge for PCI Express Gen3 core in Endpoint configuration supports up to six 32-bit BARs or three 64-bit BARs. The AXI Bridge for PCI Express in Root Port configuration supports up to two 32-bit BARs or one 64-bit BAR.

BARs can be one of two sizes. BARs 0, 2, and 4 can be either 32-bit or 64-bit addressable. BARs 1, 3, and 5 can only be 32-bit addressable and are disabled if the previous BAR is enabled as 64-bit.

- **32-bit BARs**: The address space can be as small as 4 kilobytes or as large as 2 gigabytes. Used for Memory to I/O.
- **64-bit BARs**: The address space can be as small as 4 kilobytes or as large as 256 gigabytes. Used for Memory only.

All BAR registers share these options:

- **Checkbox**: Click the checkbox to enable BAR. Deselect the checkbox to disable BAR.
- **Type**: BARs can be **Memory** apertures only. Memory BARs can be either 64-bit or 32-bit. Prefetch is enabled for 64-bit and not enabled for 32-bit. When a BAR is set as 64 bits, it uses the next BAR for the extended address space, making it inaccessible.

- **Size**: The available Size range depends on the PCIe® Device/Port Type and the Type of BAR selected. Table 4-1 lists the available BAR size ranges.

Table 4-1: **BAR Size Ranges for Device Configuration**

PCIe Device/Port Type	BAR Type	BAR Size Range
PCI Express Endpoint	32-bit Memory	4 Kilobytes - 2 Gigabytes
	64-bit Memory	4 Kilobytes - 256 Gigabytes

- **Prefetchable**: Identifies the ability of the memory space to be prefetched. This can only be enabled for 64-bit addressable bars.
- **Value**: The value assigned to BAR.
- **PCIe to AXI Translation**: This text field should be set to the appropriate value to perform the translation from the PCI Express base address to the desired AXI Base Address.

For more information about managing the Base Address Register settings, see Managing Base Address Register Settings.

Managing Base Address Register Settings

Memory indicates that the address space is defined as memory aperture. The base address register only responds to commands that access the specified address space.

Disabling Unused Resources

For best results, disable unused base address registers to conserve system resources. A base address register is disabled by deselecting unused BARs in the Vivado IDE.

PCIe Miscellaneous

The PCIe Miscellaneous screen is shown inFigure 4-6.

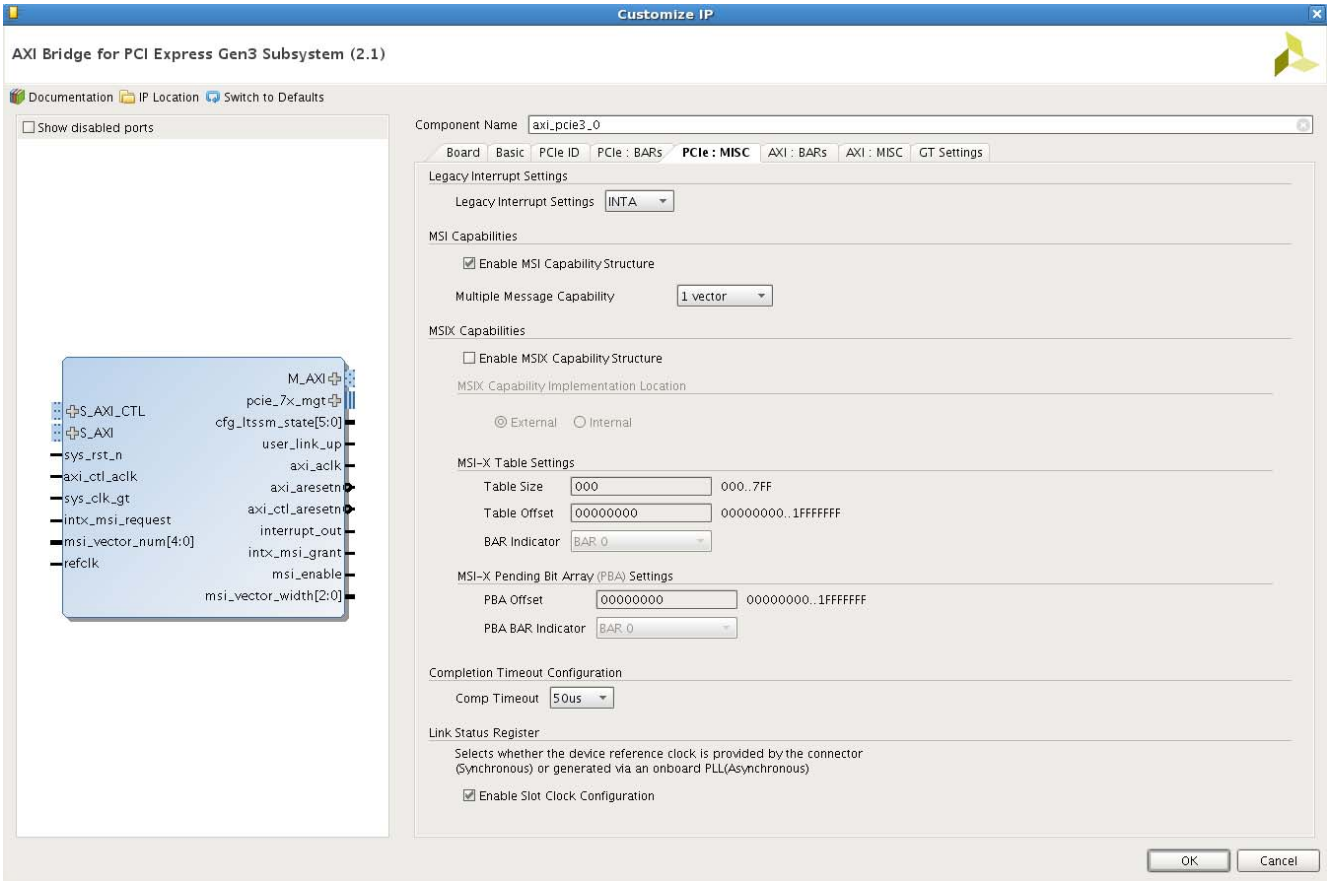

Figure 4-6: **PCIe Miscellaneous Setting**

Legacy Interrupt Settings

Indicates the usage of Legacy interrupts.The AXI Bridge for PCI Express Gen3 core implements INTA only.

Enable MSI Capability Structure

Indicates that the MSI capability structure exists. Cannot be enabled when MSI-X Capability Structure is enabled.

Enable MSIX Capability Structure

Indicates that the MSI-X capability structure exists. Cannot be enabled when MSI Capability Structure is enabled.

MSIx Table Settings

Defines the MSI-X Table structure.

- **Table Size**: Specifies the MSI-X Table size.

- **Table Offset**: Specifies the offset from the Base Address Register that points to the base of the MSI-X Table.
- **BAR Indicator**: Indicates the Base Address Register in the Configuration Space used to map the function in the MSI-X Table onto memory space. For a 64-bit Base Address Register, this indicates the lower DWORD.

MSIx Pending Bit Array (PBA) Settings

Defines the MSI-X Pending Bit Array (PBA) structure.

- **PBA Offset**: Specifies the offset from the Base Address Register that points to the base of the MSI-X PBA.
- **PBA BAR Indicator**: Indicates the Base Address Register in the Configuration Space used to map the function in the MSI-X PBA onto Memory Space.

Multiple Message Capable

Indicates the number of message signals interrupt vectors that this endpoint could request.

Completion Timeout Configuration

Indicates the completion timeout value for incoming completions due to outstanding memory read requests.

AXI Base Address Registers

The AXI Base Address Registers (BARs) screen shown inFigure 4-7 sets the AXI base address registers and the translation between AXI Memory space and PCI Express Memory space. Each BAR has a Base Address, High Address, and translation field which can be configured through the Vivado IDE.

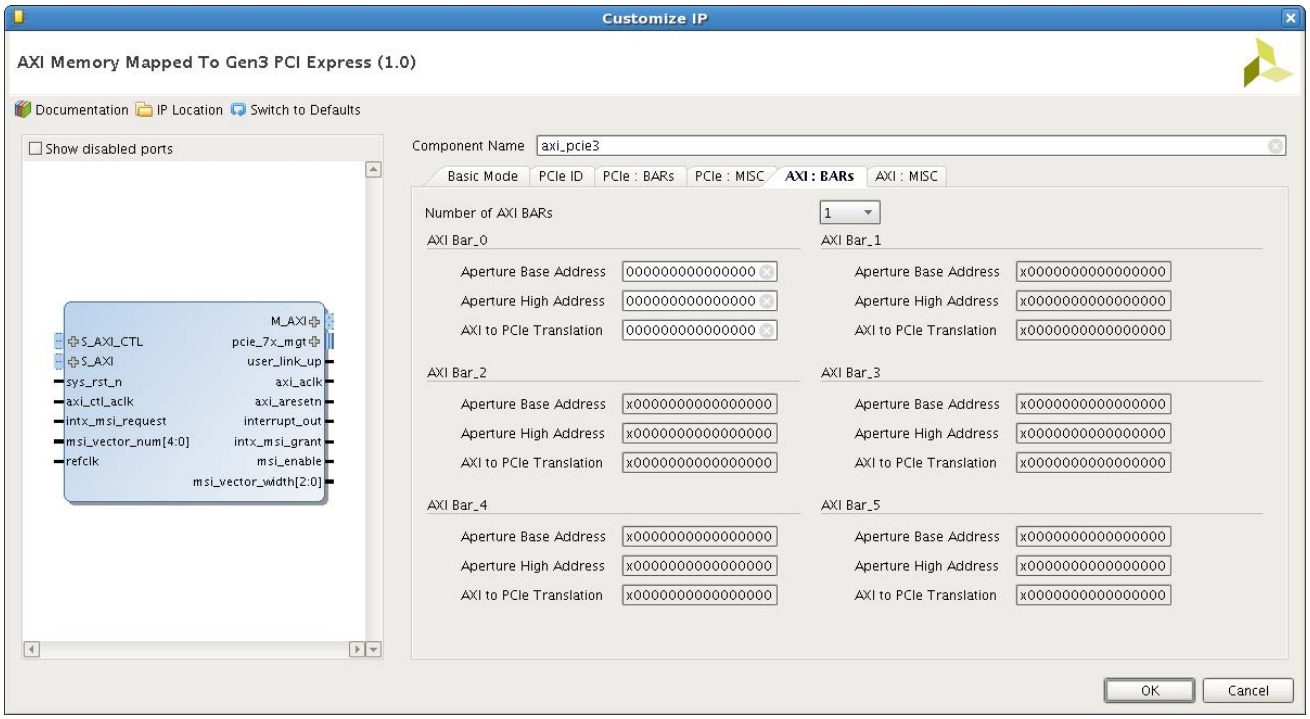

Figure 4-7: **AXI Base Address Registers**

Number of BARs

Indicates the number of AXI BARs enabled. The BARs are enabled sequentially.

Aperture Base Address

Sets the base address for the address range of BAR. You should edit this parameter to fit design requirements.

Aperture High Address

Sets the upper address threshold for the address range of BAR. You should edit this parameter to fit design requirements.

AXI to PCIe Translation

Configures the translation mapping between AXI and PCI Express address space. You should edit this parameter to fit design requirements.

AXI Miscellaneous

The AXI Miscellaneous screen shown in Figure 4-8 sets the additional AXI parameters.

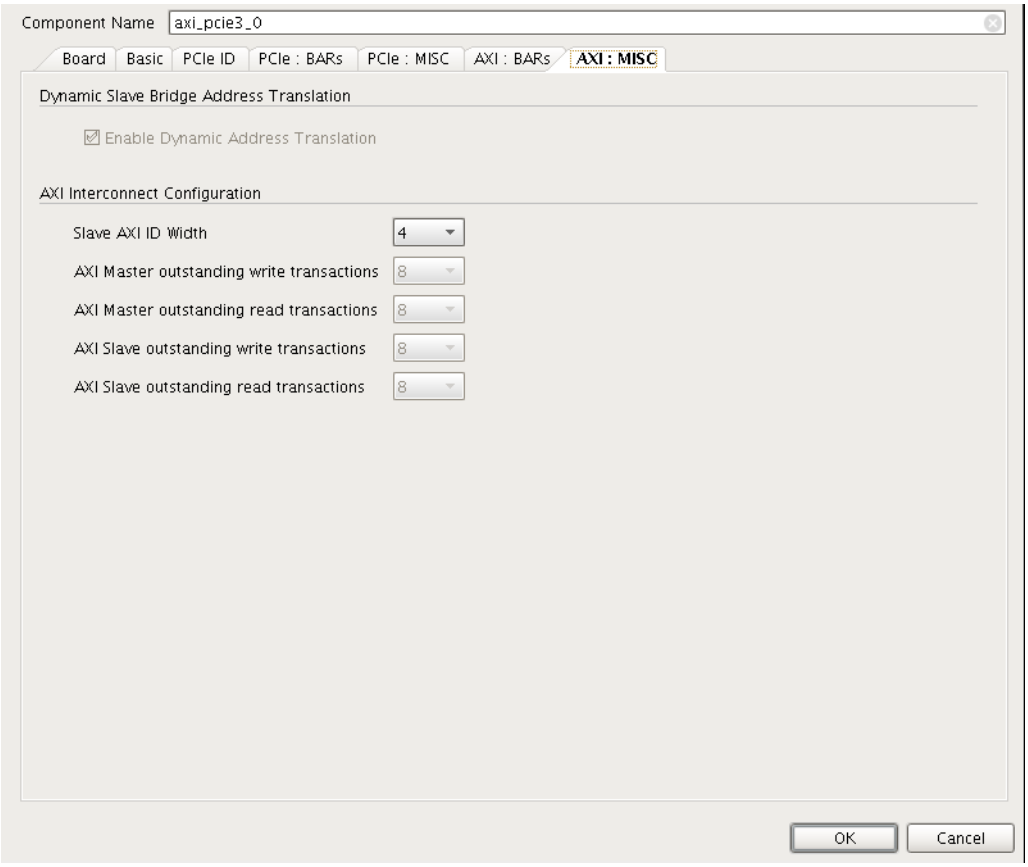

Figure 4-8: **AXI Miscellaneous Settings**

S AXI ID WIDTH

Sets the ID width for the AXI Slave Interface.

Note: Multiple IDs are not supported for AXI Master Interface. Therefore, all signals concerned with ID are not available at AXI Master Interface.

AXI Master outstanding write transactions

Indicates the number of outstanding write transactions that are allowable for the AXI Master interface.

AXI Master outstanding read transactions

Indicates the number of outstanding read transactions that are allowable for the AXI Master interface.

AXI Slave outstanding write transactions

Indicates the number of outstanding write transactions that are allowable for the AXI Slave interface.

AXI Slave outstanding read transactions

Indicates the number of outstanding read transactions that are allowable for the AXI Slave interface.

GT Settings

Figure 4-9 shows the GT Settings tab.

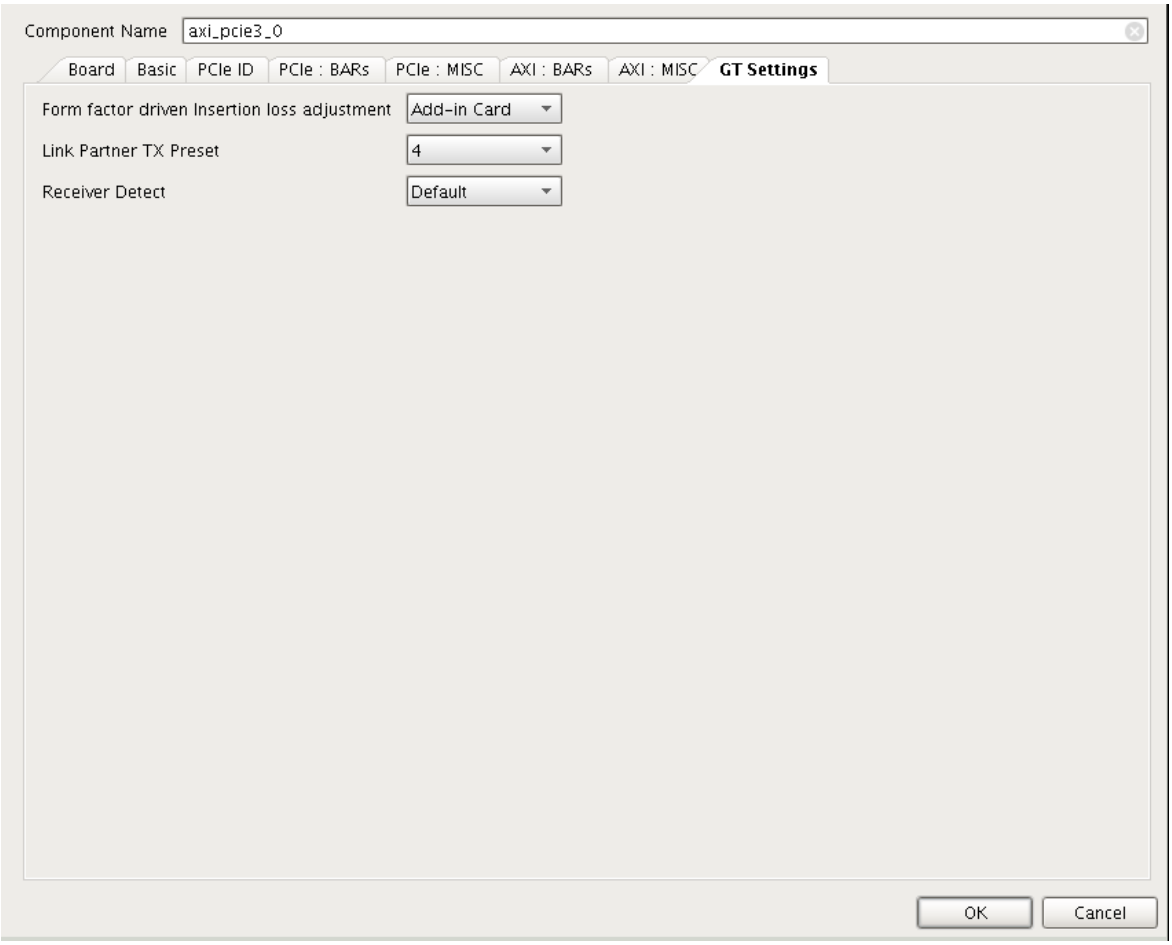

Figure 4-9: **GT Settings**

Form factor driven Insertion loss adjustment

Indicates the insertion loss profile options. There are three options provided.

1. Chip-to-Chip (5 dB)
2. Add-in Card (15 dB)
3. Backplane (20 dB)

Link Partner TX Preset

Available only when an UltraScale device is selected.

The default value is 4. It is not advisable to change the default value unless recommended by Xilinx Technical Support.

Receiver Detect

Indicates the type of Receiver Detect: **Default** or **Falling Edge**. This parameter is available in the GT Settings Tab when Advanced mode is selected. This parameter is available only for Production devices. When the **Falling Edge** option is selected, the **GT Channel DRP** Parameter in the Basic tab (in Advanced mode) is disabled. For more information on the receiver falling edge detect, see the applicable GT User Guide for your targeted device (*7 Series FPGAs GTX/GTH Transceivers User Guide* (UG476) [Ref 2], or *UltraScale Architecture GTH Transceivers User Guide* (UG576) [Ref 3]).

Output Generation

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 14].

For information regarding the example design, see Example Design Output Structure in Chapter 5.

Constraining the Core

This section contains information about constraining the core in the Vivado Design Suite.

Required Constraints

The AXI Bridge for PCI Express Gen3 core requires a clock period constraint for the reference clock[(1)] input that agrees with the REF_CLK_FREQ parameter setting. In addition, pin-placement (LOC) constraints are needed that are board/part/package specific.

See Placement Constraints for more details on the constraint paths for FPGA architectures.

Additional information on clocking can be found in the Xilinx Solution Center for PCI Express (see Solution Centers, page 86).

1. The reference clock input is `refclk` in Virtex-7 devices and `sys_clk_gt` in UltraScale devices.

System Integration

Some additional components to this system in the Vivado IP integrator can include the need to connect the MicroBlaze™ processor or Zynq® device ARM® processor peripheral to communicate with PCI Express (in addition to the AXI4-Lite register port on the PCIe bridge). The AXI Interconnect provides this capability and performs the necessarily conversions for the various AXI ports that might be connected to the AXI Interconnect IP (see *AXI to AXI Connector Data Sheet* (DS803)[Ref 11]).

The AXI Bridge for PCI Express Gen3 core can be configured with each port connection for an AXI Vivado IP integrator system topology. When instantiating the core, ensure the following bus interface tags are defined.

```
BUS_INTERFACE M_AXI
BUS_INTERFACE S_AXI
BUS_INTERFACE S_AXI_CTL
```

PCIe Clock Integration

The PCIe differential clock input in the system might need to use a differential input buffer (that is instantiated separately) from the AXI Bridge for PCI Express Gen3 core. The Vivado IP integrator automatically inserts the appropriate clock buffer.

Placement Constraints

The AXI Bridge for PCI Express Gen3 core provides a Xilinx design constraint (XDC) file for all supported PCIe, Part, and Package permutations. You can find the generated XDC file in the Sources tab of the Vivado IDE after generating the IP in the Customize IP dialog box.

For design platforms, it might be necessary to manually place and constrain the underlying blocks of the AXI Bridge for PCI Express Gen3 core. The modules to assign a LOC constraint include:

- the embedded integrated block for PCIe itself
- theGTH transceivers (for each channel)
- the PCIe differential clock input (if utilized)

The following subsection describes the example location constraints.

Location Constraints

This section highlights the LOC constraints to be specified in the XDC file for the AXI Bridge for PCI Express Gen3 core for design implementations.

For placement/path information on the integrated block for PCIe itself, the following constraint can be utilized:

```
# 7-Series Constraint
set_property LOC PCIE_X*Y* [get_cells {axi_pcie3_0_i/inst/pcie3_ip_i/inst/
pcie_top_i/pcie_7vx_i/PCIE_3_0_i}]
# Ultrascale Constraint
set_property LOC PCIE_X*Y* [get_cells {axi_pcie3_0_i/inst/pcie3_ip_i/inst/
pcie3_uscale_top_inst/pcie3_uscale_wrapper_inst/PCIE_3_1_inst}]
```

For placement/path information of the GTH transceivers, the following constraint can be utilized:

```
# 7-Series Constraint
set_property LOC GTXE2_CHANNEL_X*Y* [get_cells {axi_pcie3_0_i/inst/pcie3_ip_i/inst/
gt_top_i/pipe_wrapper_i/pipe_lane[0].gt_wrapper_i/gth_channel.gthe2_channel_i}]
# Ultrascale Constraint
set_property LOC GTXE2_CHANNEL_X*Y* [get_cells {axi_pcie3_0_i/inst/pcie3_ip_i/inst/
gt_top_i/gt_wizard.gtwizard_top_i/axi_pcie3_0_pcie3_ip_gt_i/inst/
gen_gtwizard_gthe3_top.axi_pcie3_0_pcie3_ip_gt_gtwizard_gthe3_inst/
gen_gtwizard_gthe3.gen_channel_container[1].gen_enabled_channel.gthe3_channel_wrapp
er_inst/channel_inst/
gthe3_channel_gen.gen_gthe3_channel_inst[0].GTHE3_CHANNEL_PRIM_INST}]
```

For placement/path constraints of the input PCIe differential clock source (using the example provided in System Integration), the following can be utilized:

```
set_property LOC IBUFDS_GTE2_X*Y* [get_cells {refclk_ibuf}]
```

Clock Frequencies

The AXI Memory Mapped to PCI Express Bridge supports reference clock frequencies of 100 MHz, 125 MHz, and 250 MHz and is configurable within the Vivado IDE.

Clock Management

For details, see Clocking in Chapter 3.

Clock Placement

For details, see Placement Constraints.

Banking

This section is not applicable for this IP core.

Transceiver Placement

The Transceiver primitives adjacent to the PCIe hard block should be used to aid in the place and route of the solution IP. The adjacent Transceiver banks one above or one below the desired PCIe hard block can also be used. Transceivers outside this range are not likely to meet the timing requirements for the PCI Express Solution IP and should not be used.

I/O Standard and Placement

The sys_reset_n input should be driven directly by FPGA I/O pins. The pin should be driven by the PCI Express edge connector reset signal (perstn). As described in the PCI Express specification, the PCI Express edge connector reset is driven at 3.3V. If the bank voltage for the sys_reset_n pin differs from 3.3V, and external level shifter is required to convert the 3.3V PCI Express edge connector reset (perstn) to the desired FPGA bank voltage.

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 15].
- For information regarding simulating the example design, see Simulation Design Overview in Chapter 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.*

Post-Synthesis/Post-Implementation Netlist Simulation

The AXI Bridge for PCI Express Gen3 core supports post-synthesis/post-implementation netlist functional simulations. However, some configurations do not support this feature in this release. See Table 4-2 for the configuration support of netlist functional simulations.

Note: Post-synthesis/implementation netlist *timing* simulations are not supported for any of the configurations in this release.

Table 4-2: **Configuration Support for Functional Simulation**

Configuration	Verilog	VHDL	PIPE Mode Option/ External PIPE Interface	Shared Logic in Core	Shared Logic in Example Design
Endpoint	Yes	N/A	No	N/A	

Post-Synthesis Netlist Functional Simulation

To run a post-synthesis netlist functional simulation:

1. Generate the core with required configuration
2. Open the example design and run Synthesis
3. After synthesis is completed, in the Flow Navigator, right-click the **Run Simulation** option and select **Run Post-Synthesis Functional Simulation**.

Post-Implementation Netlist Functional Simulation

To run post-implementation netlist functional simulations:

1. Complete the above steps post-synthesis netlist function simulation.
2. Run the implementation for the generated example design.
3. After implementation is completed, in the Flow Navigator, right-click the **Run Simulation** option and select **Run Post-Implementation Functional Simulation**.

Synthesis and Implementation

- For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 14].
- For information regarding implementing the example design, see Implementation Design Overview in Chapter 5.

Example Design

This chapter contains information about the example design provided in the Vivado® Design Suite.

Overview

The example simulation design for the Endpoint configuration of the AXI-PCIe block consists of two parts.

- **Root Port Model**: a test bench that generates, consumes, and checks PCI Express bus traffic
- **AXI Block RAM Controller**

Simulation Design Overview

For the simulation design, transactions are sent from the Root Port Model to the AXI Bridge for PCI Express Gen3 core configured as an Endpoint and processed inside the AXI Block RAM controller design.

Figure 5-1 illustrates the simulation design provided with the AXI Bridge for PCI Express Gen3 core.

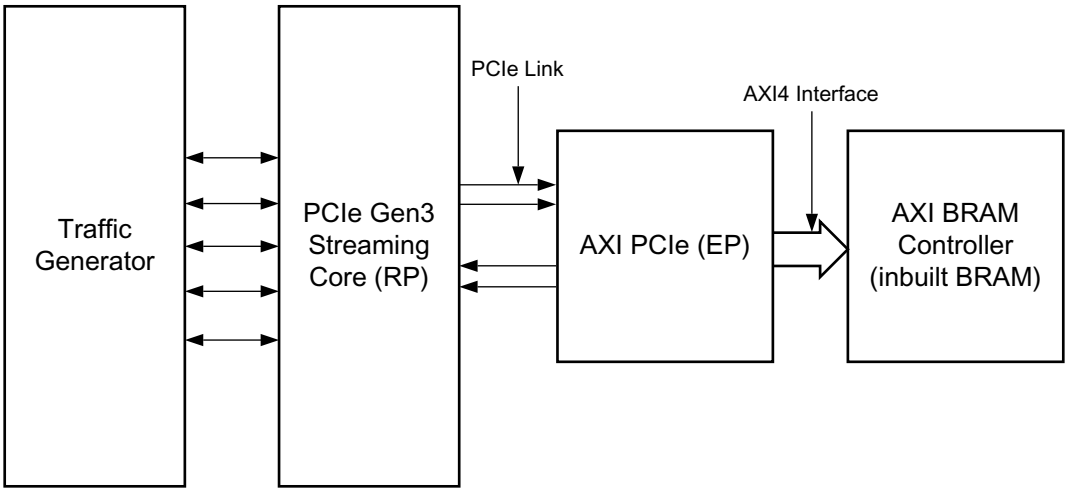

Figure 5-1: **Example Design Block Diagram**

Note: The example design supports Verilog as the target language.

Customizing and Generating the Example Design

In the Customize IP dialog box, make the following selections for the example design.

1. In the PCIE:Basics page, the example design supports only an Endpoint (EP) device.
2. The PCIE:ID defaults are supported.
3. The PCIE:BARS defaults are supported.
4. The PCIE:Misc page defaults are supported.
5. In the AXI:BARS page, default values are assigned to the Base Address, High Address, and AXI to PCIe Translation values.
6. The AXI:System page default values are supported.
7. The GT Settings page default values are supported.

Note: After customizing the core, right-click the component name, and select **Open IP Example Design**. This opens a separate example design. Simulate the core by following the steps in the next section.

Simulating the Example Design

The example design can be run in any configuration using:

- Vivado simulator
- Cadence IES Simulator
- Mentor Graphics QuestaSim

- VCS Simulator

Vivado Simulator

By defaults, the simulator is set to Vivado simulator. To run a simulation, click **Run Behavioral Simulation** in the Flow Navigator.

Cadence IES

For a Cadence IES simulation, the following steps are required:

1. In Vivado IDE, change the simulation settings as follows:
   - Target simulator: **Incisive Enterprise Simulator (IES)**
2. On the simulator tab, select **Run Simulation > Run behavioral simulation**.

Mentor Graphics QuestaSim

For a QuestaSim simulation, the following steps are required:

1. In the Vivado IDE, change the simulation settings as follows:
   - Target simulator: **QuestaSim/ModelSim**
2. On the Simulator tab, select **Run Simulation > Run behavioral simulation**.

IMPORTANT: *Due to a bug with the QuestaSim version provided with the current Vivado Design Suite, simulation fails with a Signal Abort (SIGABRT) error. To resolve the issue, use* ***QuestaSim 10.3c_1*** *instead.*

VCS Simulator

For a VCS simulation, the following steps are required:

1. In Vivado IDE, change the simulation settings as follows:
   - Target simulator: **Verilog Compiler Simulator (VCS)**
2. On the simulator tab, select **Run Simulation > Run behavioral simulation**.

Implementation Design Overview

For implementation design, the AXI Block RAM controller can be used as a scratch pad memory to write and read to Block RAM locations.

Example Design Elements

The core wrapper includes:

- An example Verilog HDL or VHDL wrapper (instantiates the cores and example design).
- A customizable demonstration test bench to simulate the example design.

Example Design Output Structure

Figure 5-2 provides the output structure of the example design.

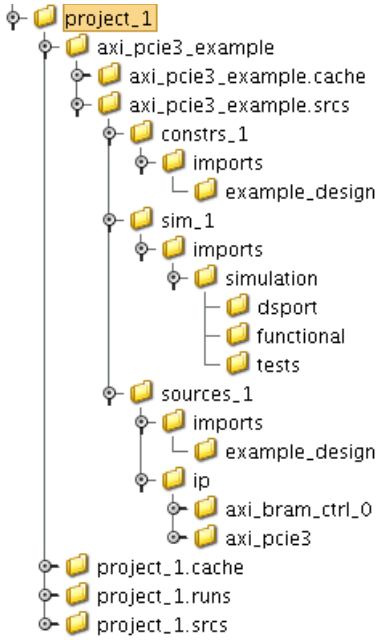

Figure 5-2: **Example Design Output Structure**

Table 5-1 provides a descriptions of the contents of the example design directories.

Table 5-1: **Example Design Structure**

Directory	Description
project_1/axi_pcie3_example	Contains all example design files.
project_1/axi_pcie3_example/ axi_pcie3_example.srcs/sources_1/imports/ example_design/	Contains the top module for the example design, `xilinx_axi_pcie3_ep.v`.

Directory	Description
project_1/axi_pcie3_example/axi_pcie3_example.srcs/sources_1/ip/axi_pcie3	Contains the XDC file based on device selected, all design files and subcores used in axi_pcie, and the top modules for simulation and synthesis.
project_1/axi_pcie3_example/axi_pcie3_example.srcs/sources_1/ip/axi_bram_ctrl_0	Contains block RAM controller files used in example design.
project_1/axi_pcie3_example/axi_pcie3_example.srcs/sim_1/imports/simulation/dsport	Contains all RP files, cgator and PIO files.
project_1/axi_pcie3_example/axi_pcie3_example.srcs/sim_1/imports/simulation/functional	Contains the test bench file.
project_1/axi_pcie3_example/axi_pcie3_example.srcs/constrs_1/imports/example_design	Contains the example design XDC file.

Test Bench

This chapter contains information about the test benches provided in the Vivado® Design Suite environment.

Root Port Model Test Bench for Endpoint

The PCI Express® Root Port Model is a robust test bench environment that provides a test program interface that can be used with the provided PIO design or with a user design. The purpose of the Root Port Model is to provide a source mechanism for generating downstream PCI Express TLP traffic to stimulate your design, and a destination mechanism for receiving upstream PCI Express TLP traffic from your design in a simulation environment.

Source code for the Root Port Model is included to provide the model for a starting point for your test bench. All the significant work for initializing the configuration space, creating TLP transactions, generating TLP logs, and providing an interface for creating and verifying tests are complete, allowing you to dedicate efforts to verifying the correct functionality of the design rather than spending time developing an Endpoint core test bench infrastructure.

The Root Port Model consists of:

- Test Programming Interface (TPI), which allows you to stimulate the Endpoint device for the model.
- Example tests that illustrate how to use the test program TPI.

Figure 1 illustrates the Root Port Model coupled with the PIO design.

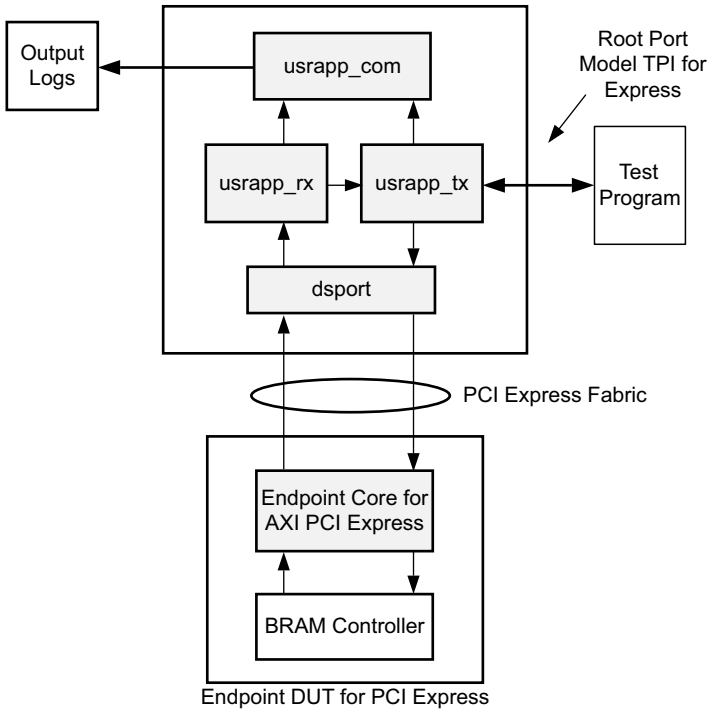

Figure 6-1: **Root Port Model for AXI_PCIE Endpoint**

Architecture

The Root Port Model consists of these blocks, illustrated in Figure 6-1:

- dsport (Root Port)
- usrapp_tx
- usrapp_rx
- usrapp_com (Verilog only)

The usrapp_tx and usrapp_rx blocks interface with the dsport block for transmission and reception of TLPs to/from the Endpoint Design Under Test (DUT). The Endpoint DUT consists of the Endpoint for AXI-PCIe® and the Block RAM controller design (displayed) or customer design.

The usrapp_tx block sends TLPs to the dsport block for transmission across the PCI Express Link to the Endpoint DUT. In turn, the Endpoint DUT device transmits TLPs across the PCI Express Link to the dsport block, which are subsequently passed to the usrapp_rx block. The dsport and core are responsible for the data link layer and physical link layer processing when communicating across the PCI Express logic. Both usrapp_tx and usrapp_rx utilize the usrapp_com block for shared functions, for example, TLP processing and log file outputting.

Transaction sequences or test programs are initiated by the usrapp_tx block to stimulate the logic interface of the Endpoint device. TLP responses from the Endpoint device are received by the usrapp_rx block. Communication between the usrapp_tx and usrapp_rx blocks allow the usrapp_tx block to verify correct behavior and act accordingly when the usrapp_rx block has received TLPs from the Endpoint device.

Simulating the Example Design

To simulate the design, see Chapter 5, Example Design.

Endpoint Model Test Bench for Root Port

The Endpoint model test bench for the AXI Bridge for PCI Express Gen3 core in Root Port configuration is a simple example test bench that connects the Configurator example design and the PCI Express Endpoint model allowing the two to operate like two devices in a physical system. Because the Configurator example design consists of logic that initializes itself and generates and consumes bus traffic, the example test bench only implements logic to monitor the operation of the system and terminate the simulation.

The Endpoint model test bench consists of:

- Verilog source code for all Endpoint model components
- PIO slave design

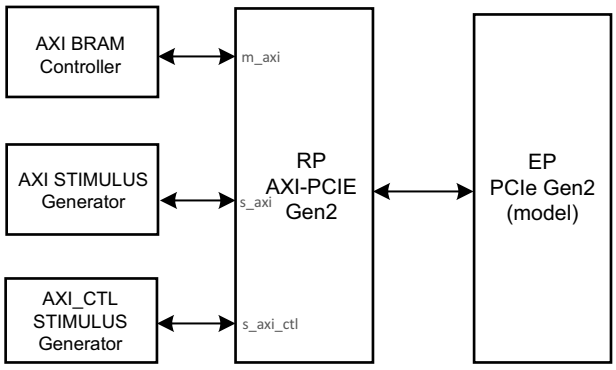

Figure 6-2: **Endpoint Model for AXI_PCIE Root Port**

Architecture

The Endpoint model consists of these blocks:

- PCI Express Endpoint (AXI Bridge for PCI Express Gen3 in Endpoint configuration) model.
- PIO slave design, consisting of:
  - pio_rx_engine
  - pio_tx_engine
  - pio_ep_mem
  - pio_to_ctrl

The pio_rx_engine and pio_tx_engine blocks interface with the ep block for reception and transmission of TLPs from/to the Root Port Design Under Test (DUT). The Root Port DUT consists of the core configured as a Root Port and the Block RAM controller along with s_axi and s_axi_ctl models to drive traffic on s_axi and s_axi_ctl.

Simulating the Example Design

To simulate the design, see Chapter 5, Example Design.

Debugging

This appendix provides information for using the resources available on the Xilinx® Support website, debug tools, and other step-by-step processes for debugging designs that use the AXI Bridge for PCI Express Gen3 core.

Finding Help on Xilinx.com

To help in the design and debug process when using the AXI Bridge for PCI Express Gen3 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.

Documentation

This product guide is the main document associated with the AXI Bridge for PCI Express Gen3 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, see the online help after installation.

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.

The PCI Express Solution Center is located at Xilinx Solution Center for PCI Express. Extensive debugging collateral is available in AR: 56802.

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 are listed below, and 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:

- the 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 AXI Bridge for PCI Express Gen3

AR: 61898

Technical Support

Xilinx provides technical support in the Xilinx Support web page for this 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 AXI Bridge for PCI Express Gen3 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 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 16].

Reference Boards

Various Xilinx development boards support the AXI Bridge for PCI Express Gen3 core. These boards can be used to prototype designs and establish that the core can communicate with the system.

- 7 series evaluation boards
  - VC709
  - VCU108

Third-Party Tools

This section describes third-party software tools that can be useful in debugging.

LSPCI (Linux)

LSPCI is available on Linux platforms and allows you to view the PCI Express® device configuration space. LSPCI is usually found in the `/sbin` directory. LSPCI displays a list of devices on the PCI buses in the system. See the LSPCI manual for all command options. Some useful commands for debugging include:

- `lspci -x -d [<vendor>]:[<device>]`

  This displays the first 64 bytes of configuration space in hexadecimal form for the device with vendor and device ID specified (omit the -d option to display information for all devices). The default Vendor/Device ID for Xilinx cores is 10EE:6012. Here is a sample of a read of the configuration space of a Xilinx device:

```
> lspci -x -d 10EE:6012
81:00.0 Memory controller: Xilinx Corporation: Unknown device 6012
00: ee 10 12 60 07 00 10 00 00 00 80 05 10 00 00 00
10: 00 00 80 fa 00 00 00 00 00 00 00 00 00 00 00 00
20: 00 00 00 00 00 00 00 00 00 00 00 00 ee 10 6f 50
30: 00 00 00 00 40 00 00 00 00 00 00 00 05 01 00 00
```

Included in this section of the configuration space are the Device ID, Vendor ID, Class Code, Status and Command, and Base Address Registers.

- `lspci -xxxx -d [<vendor>]:[<device>]`

  This displays the extended configuration space of the device. It can be useful to read the extended configuration space on the root and look for the Advanced Error Reporting (AER) registers. These registers provide more information on why the device has flagged an error (for example, it might show that a correctable error was issued because of a replay timer timeout).

- `lspci -k`

  Shows kernel drivers handling each device and kernel modules capable of handling it (works with kernel 2.6 or later).

PCItree (Windows)

PCItree can be downloaded at www.pcitree.de and allows you to view the PCI Express device configuration space and perform one DWORD memory writes and reads to the aperture.

The configuration space is displayed by default in the lower right corner when the device is selected, as shown in Figure A-1.

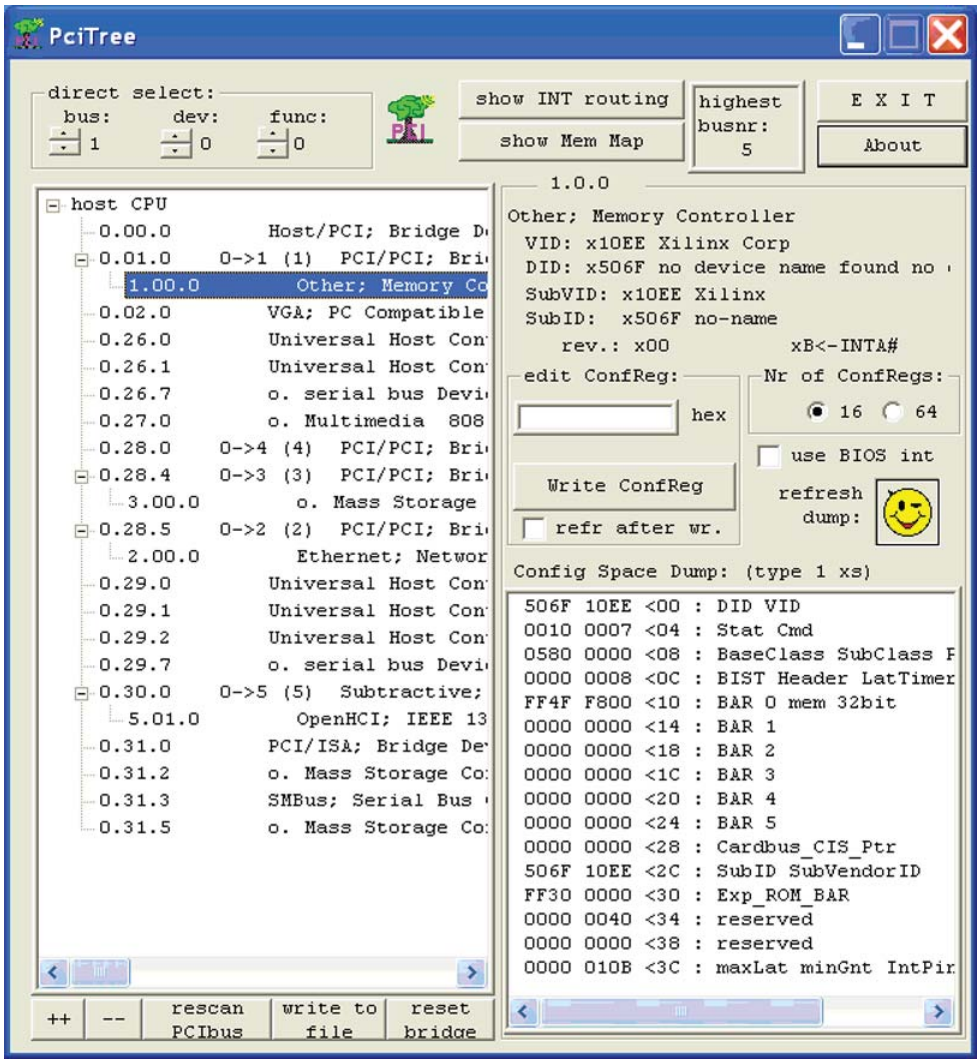

Figure A-1: **PCItree with Read of Configuration Space**

Hardware Debug

Transceiver Debug

IMPORTANT: *The ports in the Transceiver Control And Status Interface must be driven in accordance with the appropriate GT user guide. Using the input signals listed in Table A-1 and Table A-2 may result in unpredictable behavior of the IP core.*

Table A-1 describes the ports used to debug transceiver related issues when a 7 series device is targeted.

Table A-1: **Ports Used for Transceiver Debug**

Port	Direction (I/O)	Width	Description
pipe_txprbssel	I	3	PRBS input
pipe_rxprbssel	I	3	PRBS input
pipe_rxprbsforceerr	I	1	PRBS input
pipe_rxprbscntrreset	I	1	PRBS input
pipe_loopback	I	1	PIPE loopback
pipe_rxprbserr	O	1	PRBS output
pipe_rst_fsm	O		Should be examined if PIPE_RST_IDLE is stuck at 0.
pipe_qrst_fsm	O		Should be examined if PIPE_RST_IDLE is stuck at 0.
pipe_sync_fsm_tx	O		Should be examined if PIPE_RST_FSM stuck at 11'b10000000000, or PIPE_RATE_FSM stuck at 24'b000100000000000000000000.
pipe_sync_fsm_rx	O		Deprecated.
pipe_drp_fsm	O		Should be examined if PIPE_RATE_FSM is stuck at 100000000.
pipe_rst_idle	O		Wrapper is in IDLE state if PIPE_RST_IDLE is High.
pipe_qrst_idle	O		Wrapper is in IDLE state if PIPE_QRST_IDLE is High.
pipe_rate_idle	O		Wrapper is in IDLE state if PIPE_RATE_IDLE is High.
PIPE_DEBUG_0/gt_txresetdone	O		Generic debug ports to assist debug. These are generic debug ports to bring out internal PIPE Wrapper signals, such as raw GT signals. DEBUG_0 to DEBUGT_9 are intended for per lane signals. The bus width of these generic debug ports depends on the number of lanes configured in the wrapper.
PIPE_DEBUG_1/gt_rxresetdone	O		Generic debug ports to assist debug. These are generic debug ports to bring out internal PIPE Wrapper signals, such as raw GT signals. DEBUG_0 to DEBUGT_9 are intended for per lane signals.The bus width of these generic debug ports depends on the number of lanes configured in the wrapper.
PIPE_DEBUG_2/gt_phystatus	O		Generic debug ports to assist debug. These are generic debug ports to bring out internal PIPE Wrapper signals, such as raw GT signals. DEBUG_0 to DEBUGT_9 are intended for per lane signals. The bus width of these generic debug ports depends on the number of lanes configured in the wrapper.

Table A-1: **Ports Used for Transceiver Debug *(Cont'd)***

Port	Direction (I/O)	Width	Description
PIPE_DEBUG_3/gt_rxvalid	O		Generic debug ports to assist debug. These are generic debug ports to bring out internal PIPE Wrapper signals, such as raw GT signals. DEBUG_0 to DEBUGT_9 are intended for per lane signals. The bus width of these generic debug ports depends on the number of lanes configured in the wrapper.
PIPE_DEBUG_4/ gt_txphaligndone	O		Generic debug ports to assist debug. These generic debug ports bring out internal PIPE Wrapper signals, such as raw GT signals. DEBUG_0 to DEBUGT_9 are intended for per lane signals. The bus width of these generic debug ports depends on the number of lanes configured in the wrapper.
PIPE_DEBUG_5/ gt_rxphaligndone	O		Generic debug ports to assist debug. These generic debug ports bring out internal PIPE Wrapper signals, such as raw GT signals. DEBUG_0 to DEBUGT_9 are intended for per lane signals. The bus width of these generic debug ports depends on the number of lanes configured in the wrapper.
PIPE_DEBUG_6/ gt_rxcommadet	O		Generic debug ports to assist debug. These generic debug ports bring out internal PIPE Wrapper signals, such as raw GT signals. DEBUG_0 to DEBUGT_9 are intended for per lane signals. The bus width of these generic debug ports depends on the number of lanes configured in the wrapper.
PIPE_DEBUG_7/gt_rdy	O		Generic debug ports to assist debug. These generic debug ports bring out internal PIPE Wrapper signals, such as raw GT signals. DEBUG_0 to DEBUGT_9 are intended for per lane signals. The bus width of these generic debug ports depends on the number of lanes configured in the wrapper.
PIPE_DEBUG_8/ user_rx_converge	O		Generic debug ports to assist debug. These generic debug ports bring out internal PIPE Wrapper signals, such as raw GT signals. DEBUG_0 to DEBUGT_9 are intended for per lane signals. The bus width of these generic debug ports depends on the number of lanes configured in the wrapper.
PIPE_DEBUG_9/ PIPE_TXELECIDLE	O		Generic debug ports to assist debug. These generic debug ports bring out internal PIPE Wrapper signals, such as raw GT signals. DEBUG_0 to DEBUGT_9 are intended for per lane signals. The bus width of these generic debug ports depends on the number of lanes configured in the wrapper.
pipe_txinhibit	I	1	Connects to TXINHIBIT on transceiver channel primitives.

Table A-2 describes the ports used to debug transceiver related issues when an UltraScale™ device is targeted.

Table A-2: **Ports Used for Transceiver Debug**

Port	Direction (I/O)	Width	Description
gt_pcieuserratedone	I	1	Connects to PCIEUSERRATEDONE on transceiver channel primitives
gt_loopback	I	3	Connects to LOOPBACK on transceiver channel primitives
gt_txprbsforceerr	I	1	Connects to TXPRBSFORCEERR on transceiver channel primitives
gt_txinhibit	I	1	Connects to TXINHIBIT on transceiver channel primitives
gt_txprbssel	I	4	PRBS input
gt_rxprbssel	I	4	PRBS input
gt_rxprbscntreset	I	1	Connects to RXPRBSCNTRESET on transceiver channel primitives
gt_txelecidle	O	1	Connects to TXELECIDLE on transceiver channel primitives
gt_txresetdone	O	1	Connects to TXRESETDONE on transceiver channel primitives
gt_rxresetdone	O	1	Connects to RXRECCLKOUT on transceiver channel primitives
gt_rxpmaresetdone	O	1	Connects to TXPMARESETDONE on transceiver channel primitives
gt_txphaligndone	O	1	Connects to TXPHALIGNDONE of transceiver channel primitives
gt_txphinitdone	O	1	Connects to TXPHINITDONE of transceiver channel primitives
gt_txdlysresetdone	O	1	Connects to TXDLYSRESETDONE of transceiver channel primitives
gt_rxphaligndone	O	1	Connects to RXPHALIGNDONE of transceiver channel primitives
gt_rxdlysresetdone	O	1	Connects to RXDLYSRESETDONE of transceiver channel primitives
gt_rxsyncdone	O	1	Connects to RXSYNCDONE of transceiver channel primitives
gt_eyescandataerror	O	1	Connects to EYESCANDATAERROR on transceiver channel primitives
gt_rxprbserr	O	1	Connects to RXPRBSERR on transceiver channel primitives
gt_dmonitorout	O	17	Connects to DMONITOROUT on transceiver channel primitives
gt_rxcommadet	O	1	Connects to RXCOMMADETEN on transceiver channel primitives
gt_phystatus	O	1	Connects to PHYSTATUS on transceiver channel primitives
gt_rxvalid	O	1	Connects to RXVALID on transceiver channel primitives
gt_rxcdrlock	O	1	Connects to RXCDRLOCK on transceiver channel primitives
gt_pcierateidle	O	1	Connects to PCIERATEIDLE on transceiver channel primitives
gt_pcieuserratestart	O	1	Connects to PCIEUSERRATESTART on transceiver channel primitives
gt_gtpowergood	O	1	Connects to GTPOWERGOOD on transceiver channel primitives
gt_cplllock	O	1	Connects to CPLLLOCK on transceiver channel primitives
gt_rxoutclk	O	1	Connects to RXOUTCLK on transceiver channel primitives
gt_rxrecclkout	O	1	Connects to RXRECCLKOUT on transceiver channel primitives

Table A-2: **Ports Used for Transceiver Debug *(Cont'd)***

Port	Direction (I/O)	Width	Description
gt_qpll1lock	O	1	Connects to QPLL1LOCK on transceiver common primitives
gt_rxstatus	O	3	Connects to RXSTATUS on transceiver channel primitives
gt_rxbufstatus	O	3	Connects to RXBUFSTATUS on transceiver channel primitives
gt_bufgtdiv	O	9	Connects to BUFGTDIV on transceiver channel primitives
phy_txeq_ctrl	O	2	PHY TX Equalization control bits
phy_txeq_preset	O	4	PHY TX Equalization Preset bits
phy_rst_fsm	O	4	PHY RST FSM state bits
phy_txeq_fsm	O	3	PHY RX Equalization FSM state bits (Gen3)
phy_rxeq_fsm	O	3	PHY TX Equalization FSM state bits (Gen3)
phy_rst_idle	O	1	PHY is in IDLE state
phy_rrst_n	O	1	Synchronized reset generation by sys_clk
phy_prst_n	O	1	Synchronized reset generation by pipe_clk

Additional Debug Information

Additional debugging information can be found in the product guide for your device:

- *Virtex-7 FPGAs Gen3 Integrated Block for PCI Express Product Guide* (PG023) [Ref 4]
- *UltraScale Architecture Gen3 Integrated Block for PCI Express* (PG156) [Ref 5]

Interface Debug

AXI4-Lite Interfaces

Read from a register that does not have all 0s as a default to verify that the interface is functional. Output `s_axi_arready` asserts when the read address is valid and output `s_axi_rvalid` asserts when the read data/response is valid. If the interface is unresponsive ensure that the following conditions are met.

- The `axi_ctl_aclk` and `axi_ctl_aclk_out` clock inputs are connected and toggling.
- The interface is not being held in reset, and `axi_aresetn` is an active-Low reset.
- Ensure that the main core clocks are toggling and that the enables are also asserted.

- Has a simulation been run? Verify in simulation and/or a Vivado Design Suite debug feature capture that the waveform is correct for accessing the AXI4-Lite interface.

Migrating and Upgrading

This appendix contains information about upgrading to a more recent version of the IP core.

Migrating to the Vivado Design Suite

For information on migrating to the Vivado Design Suite, see *ISE to Vivado Design Suite Migration Methodology Guide (UG911)* [Ref 12].

Migrating from AXI PCIe Gen2 to AXI PCIe Gen3

Design modifications are necessary when you move from AXI PCIe Gen2 to AXI PCIe Gen3.

Parameter Changes

The following parameters have changed from the AXI PCIe Gen2 core to the AXI PCIe Gen3 core.

Table B-1: **Parameter Name Changes**

AXI PCIe Gen2 Parameter	AXI PCIe Gen3 Parameter
C_PCIEBAR_LEN_0 C_PCIEBAR_LEN_1 C_PCIEBAR_LEN_2	PF0_BAR0_APERTURE_SIZE PF0_BAR1_APERTURE_SIZE PF0_BAR2_APERTURE_SIZE PF0_BAR3_APERTURE_SIZE PF0_BAR4_APERTURE_SIZE PF0_BAR5_APERTURE_SIZE
C_SUPPORTS_NARROW_BURST	not supported
C_NO_OF_LANES	PL_LINK_CAP_MAX_LINK_WIDTH
C_DEVICE_ID	PF0_DEVICE_ID
C_VENDOR_ID	PF0_VENDOR_ID
C_CLASS_CODE	PF0_CLASS_CODE

Table B-1: **Parameter Name Changes**

AXI PCIe Gen2 Parameter	AXI PCIe Gen3 Parameter
C_REV_ID	PF0_REVISION_ID
C_REF_CLK_FREQ	REF_CLK_FREQ

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

Table B-2 shows the new parameter added in the current version of the core.

Table B-2: **Parameter Changes**

User Parameter Name	Display Name	New/ Change/ Removed	Details	Default Value
device_port_type	Device/Port Type	Change	Removed the Legacy_PCI_Express_Endpoint _device option from available options.	PCI_Express_ Endpoint_ device
tandem_mode	Tandem Configuration or Partial Reconfiguration	Removed	This parameter has been replaced by the mcap_enablement parameter	None
mcap_enablement	Tandem Configuration or Partial Reconfiguration	New	This is the new parameter added for **Tandem mode** selection. The valid values are **None**, **Tandem**, and **Tandem with Field Updates**.	None
en_ext_startup	Use an external STARTUP primitive	Removed	This is replaced by the ext_startup_primitive parameter.	False
ext_startup_primitive	Use an external STARTUP primitive	New	When this parameter unselected the STARTUP block will be generated internal to the IP when Tandem modes are selected.	False
pf0_base_class_menu	Base Class Menu	Change	Changed the default value.	Memory Controller
pf0_sub_class_interface_menu	Subclass Interface Menu	Change	Changed the default value.	Other Memory Controller

Port Changes

Table B-3 shows the new port added in the current version of the core.

Table B-3: **Port Changes**

Name	Direction	Width
common_commands_out	Out	26 Bits(1)
pipe_tx_0_sigs	Out	84 Bits(2)
pipe_tx_1_sigs	Out	84 Bits(2)
pipe_tx_2_sigs	Out	84 Bits(2)
pipe_tx_3_sigs	Out	84 Bits(2)
pipe_tx_4_sigs	Out	84 Bits(2)
pipe_tx_5_sigs	Out	84 Bits(2)
pipe_tx_6_sigs	Out	84 Bits(2)
pipe_tx_7_sigs	Out	84 Bits(2)
gt_dmonfiforeset(3)	In	(PL_LINK_CAP_MAX_LINK_WIDTH-1):0
gt_dmonitorclk(3)	In	(PL_LINK_CAP_MAX_LINK_WIDTH-1):0

Notes:

1. This signal width has changed to match common_command_in.
2. This external pipe interface signal width have changed to match pipe_rx_*_sigs.
3. This signal was added to transceiver debug and status ports interface and is applicable only for UltraScale™ variants.

Appendix C

Additional Resources and Legal Notices

Xilinx Resources

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

References

This section provides links to supplemental material useful to this document:

1. *Vivado Design Suite AXI Reference Guide* (UG1037)
2. *7 Series FPGAs GTX/GTH Transceivers User Guide* (UG476)
3. *UltraScale Architecture GTH Transceivers User Guide* (UG576)
4. *Virtex-7 FPGAs Gen3 Integrated Block for PCI Express Product Guide* (PG023)
5. *UltraScale Architecture Integrated Block for PCI Express Product Guide* (PG156)
6. *AXI Memory Mapped to PCI Express Gen2 Product Guide* (PG055)
7. *7 Series FPGAs Clocking Resources User Guide* (UG472)
8. *UltraScale Architecture Clocking Resources User Guide* (UG572)
9. AMBA AXI Protocol Specification
10. PCI-SIG Specifications
11. *AXI to AXI Connector Data Sheet* (DS803)
12. *ISE to Vivado Design Suite Migration Methodology Guide (*UG911)
13. *Vivado Design Suite User Guide: Getting Started* (UG910)
14. *Vivado Design Suite User Guide: Designing with IP* (UG896)
15. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
16. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
17. *Vivado Design Suite User Guide: Partial Reconfiguration* (UG909)

18. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
06/08/2016	2.1	• Updated Maximum Payload Size to 512 in IP Facts. • Added axi_aresetn, s_axi_ctl_awaddr[31:0], and s_axi_ctl_araddr[31:0] in Table 2-1: Top-Level Interface Signals. • Removed C_INCLUDE_BAROFFSET_REG, added pf0_msix_cap_pba_bir to pf0_msix_cap_table_size in Table 2-2: Top-Level Parameters. • Updated Note in Address Translation section. • Updated Fig. 4-6: PCIe Miscellaneous Setting and Fig. 4-9: GT Settings. • Added MSIx Table Settings and MSIx Pending Bit Array Settings section in PCIe Miscellaneous section.
04/06/2016	2.1	• Small editorial update to the Root Port Error FIFO Read Register (Offset 0x154) section. • Added the Tandem Configuration section. • New Parameter Changes and Port Changes tables for the release added to the Migrating and Updating appendix.
11/18/2015	2.0	• Updated the supported speed grades. • Updated the Limitations section in the Overview chapter. • Updated the description for m_axi_aready. • Added VCU108 to the Reference Boards section in the Debugging chapter.
09/30/2015	2.0	• Updated for the core version 2.0. • Added support for Root Port configurations. • Added MSI-X signal support. • Updated the clock and reset diagrams. • Added the Configuration Control Register. • Clarified the Reference Clock input frequency as refclk for Virtex-7 devices, and sys_clk_gt for UltraScale devices. • Updated supported values and the defaults for the following parameters: C_S_AXI_NUM_WRITE, C_S_AXI_NUM_READ, C_M_AXI_NUM_WRITE, and C_M_AXI_NUM_READ. • Added the PL_UPSTREAM_FACING bridge parameter. • Added the RX_Detect parameter. • Added the cfg_ltssm_state port. • Updated the Register Memory Map and Bridge Info Register tables. Added the Enhanced Configuration Access table (ECAM).

Date	Version	Revision
06/24/2015	1.1	• Moved performance and resource utilization data to www.xilinx.com. • Corrected C_NUM_MSI_REQ from attribute to signal. • Corrected the documented parameters: added top-level parameters, removed erroneous entries, and corrected parameter names, and descriptions. • Added the BAR Addressing section. • Corrected Example 2 (64-bit PCIe Address Mapping). • Added ports enabled with the Enable MSIX Capability Structure options selected to the Migrating and Upgrading chapter • Updated Vivado Lab Edition to Vivado Design Suite Debug Feature.
05/07/2015	1.1	• Minor editorial update: Corrected page footers.
04/01/2015	1.1	• Specified that the narrow burst feature is not supported. • To Vivado IDE description, added new GT Settings tab, and PLL Selection, CORE CLOCK Frequency, PPM Offset between receiver and transmitter, Spread Spectrum clocking, Insertion loss at Nyquist, and Link Partner TX Preset parameters. • Added post-synthesis and post-implementation netlist simulation details. • Added Transceiver Debug information. • Updated Vivado lab tools to Vivado Lab Edition.
11/19/2014	1.0	• Added UltraScale architecture placement constraint examples. • Updated the simulation procedures for Cadence Incisive Enterprise Simulator (IES), and Verilog Compiler Simulator (VCS). • Added important note regarding the recommended version of Mentor Graphics QuestaSim to use to avoid simulation failure. • Minor edits.
10/01/2014	1.0	Initial Xilinx release

Please Read: Important Legal Notices