LogiCORE IP AXI Bridge for PCI Express (v1.02.a)

DS820 January 18, 2012 **Product Specification**

Introduction

The Advanced eXtensible Interface (AXI) Root Complex/Endpoint (RC/EP) Bridge for PCI Express® is an interface between the AXI4 and PCI Express. Definitions and references are provided in this document for all of the functional modules, registers, and interfaces that are implemented in the AXI Bridge for PCI Express. Definitions are also provided for the hardware implementation and software interfaces to the AXI Bridge for PCI Express in the Field Programmable Gate Array (FPGA).

Features

- Kintex™-7, Virtex®-6 and Spartan®-6 FPGA Integrated Blocks for PCI Express
- Kintex-7 x1, x2, x4, x8 Gen 1 and x1, x2, x4 Gen 2
- Virtex-6 x1, x2, x4 Gen 1 and x1, x2 Gen 2
- Spartan-6 x1 Gen 1
- Maximum Payload Size (MPS) up to 256 Bytes
- Messaged Signaled Interrupt (MSI)
- Legacy interrupt supported
- Memory mapped AXI4 access to PCIe® space
- PCIe access to memory mapped AXI4 space
- Tracks and manages Transaction Layer Packets (TLP) completion processing
- Detects and indicates error conditions with interrupts
- Optimal AXI4 pipeline support for enhanced performance
- Compliant to Advanced RISC Machine (ARM®) Advanced Microcontroller Bus Architecture 4 (AMBA®) AXI4 spec
- Supports up to three PCIe 32-bit or 64-bit PCIe Base Address Register (BAR) as Endpoint

LogiCORE IP Facts Table					
Core Specifics					
Supported Device Family[1]	Kintex-7, Virtex-6 [5], Spartan-6 [5]				
Supported User Interfaces	AXI4				
	Resources				**Frequency**
Configuration	**LUTs**	**FFs**	**DSP Slices**	**Block RAMs**	**Max. Freq.**
Kintex-7	(see Table 31 & Table 32)				125 MHz
Virtex-6 Endpoint 64-bit	9300	6900	0	20	125 MHz
Spartan-6 Endpoint 32-bit	5600	4000	0	24	62.5 MHz
Provided with Core					
Documentation	Product Specification				
Design Files	VHSIC Hardware Description Language (VHDL)				
Example Design	Available Online [2]				
Test Bench	Not Provided				
Constraints File	User Constraints File (UCF)				
Simulation Model	Not Provided				
Supported S/W Driver[3]	Standalone and Linux				
Tested Design Tools					
Design Entry Tools	Xilinx Platform Studio (XPS)				
Simulation[4]	Mentor Graphics ModelSim				
Synthesis Tools	Xilinx Synthesis Technology (XST)				
Support					
Provided by Xilinx, Inc.					

1. For a complete list of supported derivative devices, see IDS Embedded Edition Derivative Device Support.
2. See Answer Record on www.xilinx.com.
3. Standalone driver information can be found in the EDK or SDK installation directory.
See xilinx_drivers.htm in *<install_directory>*/doc/usenglish. Linux OS and driver support information is available from http://wiki.xilinx.com.
4. For the supported version of the tool, see the ISE Design Suite 13: Release Notes Guide.
5.) Virtex-6 and Spartan-6 FPGAs support only a PCIe endpoint core configuration.

Functional Description

The AXI Bridge for PCIe Intellectual Property (IP) core provides the translation level between the AXI4 memory-mapped embedded system to the PCI Express system. The AXI Bridge for PCIe 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 is shown in Figure 1. The AXI Bridge for PCI Express contains two sections: 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 Register Block contains registers used in the AXI Bridge for PCI Express for dynamically mapping the AXI4 MM address range provided using AXIBAR parameters to an address for PCIe range.

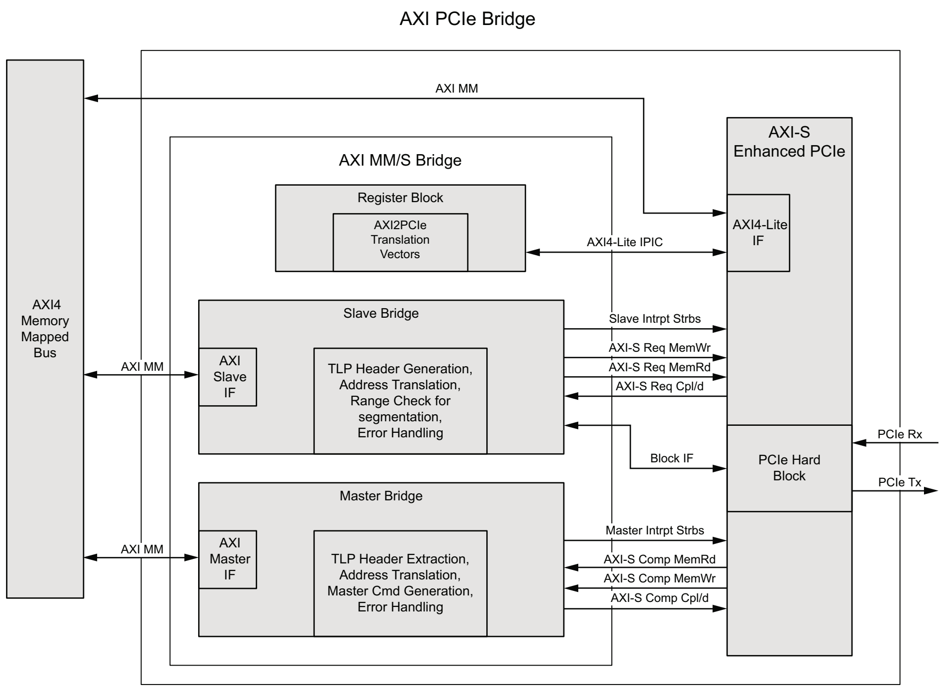

Figure 1: **AXI Bridge for PCI Express Architecture**

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 AXI Memory Mapped address domain to the domain addresses for PCIe. When a remote AXI master initiates a write transaction to the Slave Bridge, the write address and qualifiers are captured and write data is queued in a First In First Out (FIFO). These are then 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.

A second remote AXI master initiated write request write address and qualifiers can then be captured and the associated write data queued, pending the completion of the previous write TLP transfer to the integrated block for PCI Express. The resulting AXI Slave Bridge write pipeline is two-deep.

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 integrated block for PCI Express and a completion timeout timer is started. Completions received via the integrated block for PCI Express are correlated with pending read requests and read data is returned to the AXI master. The Slave bridge is capable of handling up to eight 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 is capable of supporting up to four 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 is capable of handling up to four 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.

Bridge Parameters

Because many features in the AXI Bridge for PCI Express design can be parameterized, you are able to uniquely tailor the implementation of the AXI Bridge for PCIe 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 are shown in Table 1.

Table 1: **Top Level Parameters**

Generic	Parameter Name	Description	Allowable Values	Default Value	VHDL Type
Bridge Parameters					
G1	C_FAMILY	Target FPGA Family	virtex6, spartan6, kintex7	virtex6	String
G2	C_INCLUDE_RC	Configures the AXI bridge for PCIe to be a Root Complex or an Endpoint	0 = Endpoint 1 = Root Complex (applies only for 7 series devices)	0	Integer
G3	C_COMP_TIMEOUT	Selects the Slave Bridge completion timeout counter value.	0 = 50 uS 1 = 50 mS	0	Integer
G4	C_INCLUDE_ BAROFFSET_REG	Include the registers for high-order bits to be substituted in translation in Slave Bridge	0 = exclude 1 = include	0	Integer
G5	C_SUPPORTS_ NARROW_BURST	Instantiates internal logic to support narrow burst transfers. Only enable when AXI master bridge generates narrow burst traffic.	0 = not supported 1 = supported	0	Integer

Table 1: **Top Level Parameters *(Cont'd)***

Generic	Parameter Name	Description	Allowable Values	Default Value	VHDL Type
G6	C_AXIBAR_NUM	Number of AXI address apertures that can be accessed	1- 6; 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	Integer
G7	C_AXIBAR_0	AXI BAR_0 aperture low address	Valid AXI address(1)(3)(4)	0xFFFF_FFFF	std_logic_ vector
G8	C_AXIBAR_ HIGHADDR_0	AXI BAR_0 aperture high address	Valid AXI address(1)(3)(4)	0x0000_0000	std_logic_ vector
G9	C_AXIBAR_AS_0	AXI BAR_0 address size	0 = 32 bit 1 = 64 bit	0	Integer
G10	C_AXIBAR2PCIEBAR_0	PCIe BAR to which AXI BAR_0 is mapped	Valid address for PCIe(2)	0xFFFF_FFFF	std_logic_ vector
G11	C_AXIBAR_1	AXI BAR_1 aperture low address	Valid AXI address(1)(3)(4)	0xFFFF_FFFF	std_logic_ vector
G12	C_AXIBAR_ HIGHADDR_1	AXI BAR_1 aperture high address	Valid AXI address(1)(3)(4)	0x0000_0000	std_logic_ vector
G13	C_AXIBAR_AS_1	AXI BAR_1 address size	0 = 32 bit 1 = 64 bit	0	Integer
G14	C_AXIBAR2PCIEBAR_1	PCIe BAR to which AXI BAR_1 is mapped	Valid address for PCIe(2)	0xFFFF_FFFF	std_logic_ vector
G15	C_AXIBAR_2	AXI BAR_2 aperture low address	Valid AXI address(1)(3)(4)	0xFFFF_FFFF	std_logic_ vector
G16	C_AXIBAR_ HIGHADDR_2	AXI BAR_2 aperture high address	Valid AXI address(1)(3)(4)	0x0000_0000	std_logic_ vector
G17	C_AXIBAR_AS_2	AXI BAR_2 address size	0 = 32 bit 1 = 64 bit	0	Integer
G18	C_AXIBAR2PCIEBAR_2	PCIe BAR to which AXI BAR_2 is mapped	Valid address for PCIe(2)	0xFFFF_FFFF	std_logic_ vector
G19	C_AXIBAR_3	AXI BAR_3 aperture low address	Valid AXI address(1)(3)(4)	0xFFFF_FFFF	std_logic_ vector
G20	C_AXIBAR_ HIGHADDR_3	AXI BAR_3 aperture high address	Valid AXI address(1)(3)(4)	0x0000_0000	std_logic_ vector
G21	C_AXIBAR_AS_3	AXI BAR_3 address size	0 = 32 bit 1 = 64 bit	0	Integer
G22	C_AXIBAR2PCIEBAR_3	PCIe BAR to which AXI BAR_3 is mapped	Valid address for PCIe(2)	0xFFFF_FFFF	std_logic_ vector
G23	C_AXIBAR_4	AXI BAR_4 aperture low address	Valid AXI address(1)(3)(4)	0xFFFF_FFFF	std_logic_ vector
G24	C_AXIBAR_ HIGHADDR_4	AXI BAR_4 aperture high address	Valid AXI address(1)(3)(4)	0x0000_0000	std_logic_ vector

Table 1: **Top Level Parameters *(Cont'd)***

Generic	Parameter Name	Description	Allowable Values	Default Value	VHDL Type
G25	C_AXIBAR_AS_4	AXI BAR_4 address size	0 = 32 bit 1 = 64 bit	0	Integer
G26	C_AXIBAR2PCIEBAR_4	PCIe BAR to which AXI BAR_4 is mapped	Valid address for PCIe(2)	0xFFFF_FFFF	std_logic_ vector
G27	C_AXIBAR_5	AXI BAR_5 aperture low address	Valid AXI address(1)(3)(4)	0xFFFF_FFFF	std_logic_ vector
G28	C_AXIBAR_ HIGHADDR_5	AXI BAR_5 aperture high address	Valid AXI address(1)(3)(4)	0x0000_0000	std_logic_ vector
G29	C_AXIBAR_AS_5	AXI BAR_5 address size	0 = 32 bit 1 = 64 bit	0	Integer
G30	C_AXIBAR2PCIEBAR_5	PCIe BAR to which AXI BAR_5 is mapped	Valid address for PCIe(2)	0xFFFF_FFFF	std_logic_ vector
G31	C_PCIEBAR_NUM	Number of address for PCIe apertures that can be accessed	1- 3; 1 = BAR_0 enabled 2 = BAR_0, BAR_1 enabled 3 = BAR_0, BAR_1, BAR_2 enabled	3	Integer
G32	C_PCIEBAR_AS	Configures PCIEBAR aperture width to be 32 bits wide or 64 bits wide	0 = Generates three 32-bit PCIEBAR address apertures. 32 bit BAR example: PCIEBAR_0 is 32 bits PCIEBAR_1 is 32 bits PCIEBAR_2 is 32 bits 1 = Generates three 64 bit PCIEBAR address apertures. 64-bit BAR example: PCIEBAR_0 and PCIEBAR_1 concatenate to comprise 64 bit PCIEBAR_0. PCIEBAR_2 and PCIEBAR_3 concatenate to comprise 64-bit PCIEBAR_1. PCIEBAR_4 and PCIEBAR_5 concatenate to comprise 64-bit PCIEBAR_2	1	Integer
G33	C_PCIEBAR_LEN_0	Power of 2 in the size of bytes of PCIE BAR_0 space	13-31	16	Integer
G34	C_PCIEBAR2AXIBAR_0	AXI BAR to which PCIE BAR_0 is mapped	Valid AXI address	0x0000_0000	std_logic_ vector
G35	C_PCIEBAR_LEN_1	Power of 2 in the size of bytes of PCIE BAR_1 space	13-31	16	Integer
G36	C_PCIEBAR2AXIBAR_1	AXI BAR to which PCIE BAR_1 is mapped	Valid AXI address	0x0000_0000	std_logic_ vector

Table 1: **Top Level Parameters *(Cont'd)***

Generic	Parameter Name	Description	Allowable Values	Default Value	VHDL Type
G37	C_PCIEBAR_LEN_2	Power of 2 in the size of bytes of PCIE BAR_2 space	13-31	16	Integer
G38	C_PCIEBAR2AXIBAR_2	AXI BAR to which PCIE BAR_2 is mapped	Valid AXI address	0x0000_0000	std_logic_ vector
AXI4-Lite Parameters					
G39	C_BASEADDR	Device base address	Valid AXI address	0xFFFF_FFFF	std_logic_ vector
G40	C_HIGHADDR	Device high address	Valid AXI address	0x0000_0000	std_logic_ vector
	C_S_AXI_CTL_ PROTOCOL	AXI4-Lite port connection definition to AXI Interconnect and EDK system	AXI4LITE	AXI4LITE	string
Core for PCIe Configuration Parameters					
G41	C_NO_OF_LANES	Number of PCIe Lanes	1: Spartan-6 1, 2, 4: Virtex-6 1, 2, 4, 8: 7 series	1	integer
G42	C_DEVICE_ID	Device ID	16-bit vector	0x0000	std_logic_ vector
G43	C_VENDOR_ID	Vendor ID	16-bit vector	0x0000	std_logic_ vector
G44	C_CLASS_CODE	Class Code	24-bit vector	0x00_0000	std_logic_ vector
G45	C_REV_ID	Rev ID	8-bit vector	0x00	std_logic_ vector
G46	C_SUBSYSTEM_ID	Subsystem ID	16-bit vector	0x0000	std_logic_ vector
G47	C_SUBSYSTEM_ VENDOR_ID	Subsystem Vendor ID	16-bit vector	0x0000	std_logic_ vector
	C_PCIE_USE_MODE	Specifies PCIe use mode for underlying serial transceiver wrapper usage/configuration (specific only to 7 series)	1.0 = for Kintex-7 325T IES (initial ES) silicon 1.1 = for Virtex-7 485T IES (initial ES) silicon 3.0 = for Kintex-7 325T GES (general ES) silicon	1.0	string
G48	C_PCIE_CAP_SLOT_ IMPLEMENTED	PCIE Capabilities Register Slot Implemented	0 = Not add-in card slot 1 = Downstream port is connected to add-in card slot (valid only for Root Complex)	0	integer
G49	C_REF_CLK_FREQ	REFCLK input Frequency	0 = 100 MHz 1 = 125 MHz - Spartan-6 only 2 = 250 MHz - Virtex-6 or 7 series only	0	integer

Table 1: **Top Level Parameters *(Cont'd)***

Generic	Parameter Name	Description	Allowable Values	Default Value	VHDL Type
Memory Mapped AXI4 Parameters					
G50	C_M_AXI_DATA_ WIDTH	AXI Master Bus Data width	32: Spartan-6 only 64: Virtex-6 or 7 series only 128: 7 series only	64	integer
G51	C_M_AXI_ADDR_ WIDTH	AXI Master Bus Address width	32	32	integer
G52	C_S_AXI_ID_WIDTH	AXI Slave Bus ID width	4	4	integer
G53	C_S_AXI_DATA_ WIDTH	AXI Slave Bus Data width	32: Spartan-6 only 64: Virtex-6 or 7 series only 128: 7 series only	64	integer
G54	C_S_AXI_ADDR_ WIDTH	AXI Slave Bus Address width	32	32	integer
	C_M_AXI_PROTOCOL	Protocol definition for M_AXI (Master Bridge) port on AXI Interconnect in EDK system	AXI4	AXI4	string
	C_S_AXI_PROTOCOL	Protocol definition for S_AXI (Slave Bridge) port on AXI Interconnect in EDK system.	AXI4	AXI4	string
G55	C_MAX_LINK_ SPEED	Maximum PCIe link speed supported	0 = 2.5 GT/s - Spartan-6, Virtex-6, or 7 series 1 = 5.0 GT/s - Virtex-6 or 7 series only	0	integer
G56	C_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	integer
AXI4 Interconnect Parameters					
G57	C_INTERCONNECT_S_ AXI_WRITE_ ACCEPTANCE (5)	AXI Interconnect Slave Port Write Pipeline Depth	1 = Only one active AXI AWADDR can be accepted in the AXI slave bridge for PCIe, 2 = Maximum of two active AXI AWADDR values can be stored in AXI slave bridge for PCIe.	2	integer
G58	C_INTERCONNECT_S_ AXI_READ_ ACCEPTANCE (5)	AXI Interconnect Slave Port Read Pipeline Depth	1 = Only one active AXI ARADDR can be accepted in AXI slave bridge PCIe 2, 4, 8 = Size of pipeline for active AXI ARADDR values to be stored in AXI slave bridge PCIe.	8	integer
G59	C_INTERCONNECT_M _AXI_WRITE_ISSUING (5)	AXI Interconnect Master Bridge write address issue depth	1, 2, 4 = Number of actively issued AXI AWADDR values on the AXI Interconnect to the target slave device(s).	4	integer

Table 1: **Top Level Parameters *(Cont'd)***

Generic	Parameter Name	Description	Allowable Values	Default Value	VHDL Type
G60	C_INTERCONNECT_M_AXI_READ_ISSUING [(5)]	AXI Interconnect Master Bridge read address issue depth	1, 2, 4 = Number of actively issued AXI ARADDR values on the AXI Interconnect to the target slave device(s).	4	integer

1. This is a 32-bit address.
2. The width of this should match the address size (C_AXIBAR_AS) for this BAR.
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. The address value is a 32-bit AXI address.
4. The difference between C_AXIBAR_n and C_AXIBAR_HIGHADDR_n must be less than or equal to 0x7FFF_FFFF and greater than or equal to 0x0000_1FFF.
5. It is not recommended to edit these default values on the AXI bridge for PCIe IP unless resource utilization needs to be reduced which impacts the AXI bridge performance.

Parameter Dependencies

Table 2: **Parameter Dependencies**

Generic	Parameter	Affects	Depends	Description
Bridge Parameters				
G1	C_FAMILY	G2, G41, G49, G55		
G2	C_INCLUDE_RC		G1	Meaningful only if G1 = Kintex-7. Spartan-6 and Virtex-6 are EP only.
G3	C_COMP_TIMEOUT			
G4	C_INCLUDE_BAROFFSET_REG	G10, G14, G18, G22, G26, G30	G6	If G4 = 1 then G10, G14, G18, G22, G26 and G30 have no meaning. The number of registers included is set by G6.
G5	C_SUPPORTS_NARROW_BURST			
G6	C_AXIBAR_NUM	G4, G7 - G30		If G6 = 1, then G7 - G10 are enabled. If G6 = 2, then G7 - G14 are enabled. If G6 = 3, then G7 - G18 are enabled. If G6 = 4, then G7 - G22 are enabled. If G6 = 5, then G7 - G26 are enabled. If G6 = 6, then G7 - G30 are enabled.
G7	C_AXIBAR_0	G8	G6, G8	G7 and G8 define the range in AXI memory space that is responded to by this device (AXI BAR)
G8	C_AXIBAR_HIGHADDR_0	G7	G6, G7	G7 and G8 define the range in AXI memory space that is responded to by this device (AXI BAR)
G9	C_AXIBAR_AS_0		G6	
G10	C_AXIBAR2PCIEBAR_0		G4, G6	Meaningful if G4 = 1.
G11	C_AXIBAR_1	G12	G12	G11 and G12 define the range in AXI-memory space that is responded to by this device (AXIBAR)

Table 2: **Parameter Dependencies *(Cont'd)***

Generic	Parameter	Affects	Depends	Description
G12	C_AXIBAR_HIGHADDR_1	G11	G6, G11	G11 and G12 define the range in AXI-memory space that is responded to by this device (AXIBAR)
G13	C_AXIBAR_AS_1		G6	
G14	C_AXIBAR2PCIEBAR_1		G4, G6	Meaningful if G4 = 1.
G15	C_AXIBAR_2	G16	G16	G15 and G16 define the range in AXI-memory space that is responded to by this device (AXIBAR)
G16	C_AXIBAR_HIGHADDR_2	G15	G6, G15	G15 and G16 define the range in AXI-memory space that is responded to by this device (AXIBAR)
G17	C_AXIBAR_AS_2		G6	
G18	C_AXIBAR2PCIEBAR_2		G4, G6	Meaningful if G4 = 1.
G19	C_AXIBAR_3	G20	G20	G19 and G20 define the range in AXI-memory space that is responded to by this device (AXIBAR)
G20	C_AXIBAR_HIGHADDR_3	G19	G6, G19	G19 and G20 define the range in AXI-memory space that is responded to by this device (AXIBAR)
G21	C_AXIBAR_AS_3		G6	
G22	C_AXIBAR2PCIEBAR_3		G4, G6	Meaningful if G4 = 1.
G23	C_AXIBAR_4	G24	G24	G23 and G24 define the range in AXI-memory space that is responded to by this device (AXIBAR)
G24	C_AXIBAR_HIGHADDR_4	G23	G6, G23	G23 and G24 define the range in AXI-memory space that is responded to by this device (AXIBAR)
G25	C_AXIBAR_AS_4		G6	
G26	C_AXIBAR2PCIEBAR_4		G4, G6	Meaningful if G4 = 1.
G27	C_AXIBAR_5	G28	G28	G27 and G28 define the range in AXI-memory space that is responded to by this device (AXIBAR)
G28	C_AXIBAR_HIGHADDR_5	G27	G6, G27	G27 and G28 define the range in AXI-memory space that is responded to by this device (AXIBAR)
G29	C_AXIBAR_AS_5		G6	
G30	C_AXIBAR2PCIEBAR_5		G4, G6	Meaningful if G4 = 1.
G31	C_PCIEBAR_NUM	G33-G88		If G31 = 1, then G32, G33 are enabled. If G31 = 2, then G32 - G36 are enabled. If G31 = 3, then G32 - G38 are enabled
G32	C_PCIEBAR_AS			
G33	C_PCIEBAR_LEN_0	G34	G31	
G34	C_PCIEBAR2AXIBAR_0		G31, G33	Only the high-order bits above the length defined by G33 are meaningful.
G35	C_PCIEBAR_LEN_1	G36	G31	

Table 2: **Parameter Dependencies *(Cont'd)***

Generic	Parameter	Affects	Depends	Description
G36	C_PCIEBAR2AXIBAR_1		G31, G35	Only the high-order bits above the length defined by G35 are meaningful.
G37	C_PCIEBAR_LEN_2	G38	G31	
G38	C_PCIEBAR2AXIBAR_2		G31, G37	Only the high-order bits above the length defined by G37 are meaningful.
AXI4-Lite Parameters				
G39	C_BASEADDR			
G40	C_HIGHADDR			
Core for PCIe Configuration Parameters				
G41	C_NO_OF_LANES		G1, G50, G53	If G1 = Spartan-6 then G41 = 1 only. If G1 = Virtex-6 then G41 = 1, 2 or 4. If G1 = Kintex-7 & G50 = G53 = 64, then G41 = 1, 2, or 4 (Gen 1) or G41 = 1 or 2 (Gen 2). If G1 = Kintex-7 & G50 = G53 = 128, then G41 = 1, 2, 4, or 8 (Gen 1) or G41 = 1, 2, or 4 (Gen 2). Spartan-6 is a fixed x1 EP.
G42	C_DEVICE_ID			
G43	C_VENDOR_ID			
G44	C_CLASS_CODE			
G45	C_REV_ID			
G46	C_SUBSYSTEM_ID			
G47	C_SUBSYSTEM_VENDOR_ID			
G48	C_PCIE_CAP_SLOT_ IMPLEMENTED		G2	If G2 = 0 G48 is not meaningful
G49	C_REF_CLK_FREQ		G1	If G1 = Spartan-6, G49 must be = 0 or 1. If G1 = Virtex-6, G49 must be = 0 or 2.
Memory Mapped AXI4 Bus Parameters				
G50	C_M_AXI_DATA_WIDTH	G53	G1, G41, G53	G50 must be equal to G53
G51	C_M_AXI_ADDR_WIDTH	G54	G54	G51 must be equal to G54
G52	C_S_AXI_ID_WIDTH			
G53	C_S_AXI_DATA_WIDTH	G50	G1, G41, G50	G53 must be equal to G50
G54	C_S_AXI_ADDR_WIDTH	G51	G51	G54 must be equal to G51
G55	C_MAX_LINK_SPEED		G1	If G1 = Spartan-6, G55 must be = 0.
G56	C_INTERRUPT_PIN			

Top-level Interface Signals

The interface signals for the AXI Bridge for PCI Express are described in Table 3.

Table 3: **Top Level Interface Signals**

Signal Name	I/O	Description
Global Signals		
AXI_ARESETN	I	Global reset signal for AXI Interfaces
AXI_ACLK	I	Global clock signal for AXI Interfaces
AXI_ACLK_OUT	O	PCIe derived clock output for AXI_ACLK
AXI_CTL_ACLK	I	Global clock signal for AXI CTL Interface
AXI_CTL_ACLK_OUT	O	PCIe derived clock output for AXI_CTL_ACLK
MMCM_LOCK	O	AXI_ACLK_OUT from axi_enhanced_core for PCIe is stable
INTERRUPT_OUT	O	Interrupt signal
AXI Slave Interface		
S_AXI_AWID[C_S_AXI_ID_WIDTH-1:0]	I	Slave write address ID
S_AXI_AWADDR[C_S_AXI_ADDR_WIDTH-1:0]	I	Slave address write
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[C_S_AXI_DATA_WIDTH-1:0]	I	Slave write data
S_AXI_WSTRB[C_S_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_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[C_S_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

Table 3: **Top Level Interface Signals *(Cont'd)***

Signal Name	I/O	Description
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[C_S_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
AXI Master Interface		
M_AXI_AWADDR[C_M_AXI_ADDR_WIDTH-1:0]	O	Master address write
M_AXI_AWLEN[7:0]	O	Master write burst length
M_AXI_AWSIZE[2:0]	O	Master write burst size
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[C_M_AXI_DATA_WIDTH-1:0]	O	Master write data
M_AXI_WSTRB[C_M_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_BID	I	Master response ID
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[C_M_AXI_ADDR_WIDTH-1:0]	O	Master read address
M_AXI_ARLEN[7:0]	O	Master read burst length
M_AXI_ARSIZE[2:0]	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
M_AXI_RID[3:0]	I	Master read ID tag
M_AXI_RDATA[C_M_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

Table 3: **Top Level Interface Signals *(Cont'd)***

Signal Name	I/O	Description
M_AXI_RREADY	O	Master read ready
AXI4-Lite Control Interface		
S_AXI_CTL_AWADDR[31:0]	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
S_AX_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
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
PCIe Interface		
REFCLK	I	PCIe Reference Clock
INTX_MSI_Request	I	Legacy Interrupt Input (see C_INTERRUPT_PIN) when MSI_enable = '0'. Initiates a MSI write request when MSI_enable = '1'.
MSI_enable	O	Indicates when MSI is enabled
PCI_EXP_RXP[C_NO_OF_LANES-1 : 0]	I	PCIe RX serial interface
PCI_EXP_RXN[C_NO_OF_LANES-1 : 0]	I	PCIe RX serial interface
PCI_EXP_TXP[C_NO_OF_LANES-1 : 0]	O	PCIe TX serial interface
PCI_EXP_TXN[C_NO_OF_LANES-1 :0]	O	PCIe TX serial interface

Memory Map

The memory map shown in Table 4 shows the address mapping for the AXI Bridge for PCI Express. These registers are described in more detail in the following section. All registers are accessed via the AXI4-Lite Control Interface and are offset from C_BASEADDR. During a reset, all registers return to default values.

Table 4: **Register Memory Map**

Accessibility	Offset	Contents	Location
RO - EP, R/W - RC	0x000 - 0x124	PCIe Configuration Space Header	Part of integrated PCIe configuration space.
RO	0x128	Vendor-Specific Enhanced Capability (VSEC) Capability	VSEC of integrated PCIe configuration space.
RO	0x12C	VSEC Header	
RO	0x130	Bridge Info	AXI bridge defined memory mapped register space.
RO - EP, R/W - RC	0x134	Bridge Status and Control	
R/W	0x138	Interrupt Decode	
R/W	0x13C	Interrupt Mask	
RO - EP, R/W - RC	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 - 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. For Spartan-6 device configurations, this area is read-only. For Virtex-6 device endpoint configurations, this area is read-only. For 7 series device configurations, this area is read-only when configured as an endpoint. Writes are permitted for some registers when a 7 series device is configured as a root complex. 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.

VSEC Capability Register (Offset 0x128)

The VSEC structure allows the memory space of the core to appear as though it is a part of the underlying Integrated Block for 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).

Table 5: **VSEC Capability Register**

Bit(s)	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. Hard-coded to 0x000B.
19:16	Capability Version	RO	0x1	Version of this capability structure. Hard-coded to 0x1.
31:20	Next Capability Offset	RO	0x200	Offset to next capability. Hard-coded to 0x0200.

VSEC Header Register (Offset 0x12C)

The VSEC header provides a unique (within a given vendor) identifier for the layout and contents of the VSEC structure, as well as its revision and length.

Table 6: **VSEC Header Register**

Bit(s)	Name	Core Access	Reset Value	Description
15:0	VSEC ID	RO	0x0001	ID value uniquely identifying the nature and format of this VSEC structure.
19:16	VSEC REV	RO	0	Version of this capability structure. Hard-coded to 0h.
31:20	VSEC Length	RO	0x038	Length of the entire VSEC Capability structure, in bytes, including the VSEC Capability register. Hard-coded to 0x038 (56 decimal).

Bridge Info Register (Offset 0x130)

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

Table 7: **Bridge Info Register**

Bit(s)	Name	Core Access	Reset Value	Description
0	Gen2 Capable	RO	0	If set, indicates the link is Gen2 capable. Underlying Integrated Block and Link partner support PCIe Gen2 speed.
1	Root Port Present	RO	0	Indicates 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 underlying Integrated Block is upconfig capable when this bit is set.

Table 7: **Bridge Info Register *(Cont'd)***

Bit(s)	Name	Core Access	Reset Value	Description
15:3	Reserved	RO	0	Reserved
18:16	ECAM Size	RO	0	Size of Enhanced Configuration Access Mechanism (ECAM) Bus Number field, in number of bits. If ECAM window is present, value is between 1 and 8. If not present, value is 0. Total address bits dedicated to ECAM window is 20+(ECAM Size).
31:19	Reserved	RO	0	Reserved

Bridge Status and Control Register (Offset 0x134)

The Bridge Status and Control register 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. For Spartan-6 devices, this register is not used and the contents are hard-wired to 0.

Table 8: **Bridge Status and Control Register**

Bit(s)	Name	Core Access	Reset Value	Description
0	ECAM Busy	RO	0	Indicates an ECAM access is in progress (waiting for completion).
7:1	Reserved	RO	0	Reserved
8	Global Interrupt Disable	RW	0	When set, disables interrupt line from being asserted. Does not prevent bits in Interrupt Decode register from being set.
15:9	Reserved	RO	0	Reserved
16	RW1C as RW	RW	0	When set, allows writing to core registers which are normally RW1C. Hard-wired to zero for Spartan-6 device cores.
17	RO as RW	RW	0	When set, allows writing to certain registers which are normally RO. (Only supported for Kintex-7 FPGA cores.)
31:18	Reserved	RO	0	Reserved

Interrupt Decode Register (Offset 0x138)

The Interrupt Decode register 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 to any bit of the Interrupt Decode register clears that bit.

Note: 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 9: **Interrupt Decode Register**

Bit(s)	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	ECRC Error	RW1C	0	Indicates Received packet failed ECRC check. (Only applicable to Kintex-7 FPGA cores.)
2	Streaming Error	RW1C	0	Indicates a gap was encountered in a streamed packet on the Tx interface (RW, RR, or CC).

Table 9: **Interrupt Decode Register *(Cont'd)***

Bit(s)	Name	Core Access	Reset Value	Description
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 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)
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 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 10 describes the Interrupt Mask register bits and values.

Table 10: **Interrupt Mask Register**

Bit(s)	Name	Core Access	Reset Value	Description
0	Link Down	RW	0	Enables interrupts for Link Down events when bit is set.
1	ECRC Error	RW	0	Enables interrupts for ECRC Error events when bit is set. (Only writable for EP configurations, otherwise = ‘0’)
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.

Table 10: **Interrupt Mask Register *(Cont'd)***

Bit(s)	Name	Core Access	Reset Value	Description
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 11).

Table 11: **Bus Location Register**

Bit(s)	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 Complex.
15:8	Bus Number	RO	0	Bus number of port for PCIe. For Endpoint, this register is RO and is set by the Root Complex.
23:16	Port Number	RW	0	Sets the Port number field of the Link Capabilities register. Not supported for Spartan-6 devices.
31:24	Reserved	RO	0	Reserved

PHY Status/Control Register (Offset 0x144)

The PHY Status/Control register (described in Table 12) provides the status of the current PHY state, as well as control of speed and rate switching for Gen2-capable cores.

Table 12: **PHY Status/Control Register**

Bit(s)	Name	Core Access	Reset Value	Description
0	Link Rate	RO	0	Reports the current link rate. 0b = 2.5 GT/s, 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	Lane Reversal	RO	0	Reports the current lane reversal mode. 00b = No reversal, 01b = Lanes 1:0 reversed, 10b = Lanes 3:0 reversed, 11b = Lanes 7:0 reversed.
11	Link Up	RO	0	Reports the current PHY Link-up state. 1b = Link up, 0b = Link down.
15:12	Reserved	RO	0	Reserved

Table 12: **PHY Status/Control Register *(Cont'd)***

Bit(s)	Name	Core Access	Reset Value	Description
17:16	Directed Link Width	RW	0	Specifies completer link width for a directed link change operation. Only acted on when Directed Link Change specifies a width change. 00b = x1, 01b = x2, 10b = x4, 11b = x8. (RO for Spartan-6 FPGA cores, hardwired to 0)
18	Directed Link Speed	RW	0	Specifies completer link speed for a directed link change operation. Only acted on when Directed Link Change specifies a speed change. 0b = 2.5 GT/s, 1b = 5.0 GT/s. (RO for Spartan-6 FPGA cores, hardwired to 0)
19	Directed Link Autonomous	RW	0	Specifies link reliability or autonomous for directed link change operation. 0b = link reliability, 1b = autonomous. (RO for Spartan-6 FPGA cores, hardwired to 0)
21:20	Directed Link Change	RW	0	Directs LTSSM to initiate a link width and/or speed change. 00b = no change, 01b = force link width, 10b = force link speed, 11b = force link width & speed. (RO for Spartan-6 FPGA cores, hardwired to 0)
31:22	Reserved	RO	0	Reserved

Root Port Status/Control Register (Offset 0x148)

The Root Port Status/Control register provides access to 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 13).

Table 13: **Root Port Status/Control Register**

Bit(s)	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. RC Software needs to write 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.

Table 13: **Root Port Status/Control Register *(Cont'd)***

Bit(s)	Name	Core Access	Reset Value	Description
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
27:20	Completion Timeout	RW	0	Sets the timeout counter size for Completion Timeouts.
31:28	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 14). For EP configurations, read returns zero.

Table 14: **Root Port MSI Base Register 1**

Bit(s)	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 15) 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, read returns zero.

Table 15: **Root Port MSI Base Register 2**

Bit(s)	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 queued error (Correctable/Non-fatal/Fatal) messages. Data from each read follows the format shown in Table 16. 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 16: **Root Port Error FIFO Read Register**

Bit(s)	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 is 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 17. 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 17: **Root Port Interrupt FIFO Read Register 1**

Bit(s)	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 18. 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 18: **Root Port Interrupt FIFO Read Register 2**

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

VSEC Capability Register 2 (Offset 0x200)

The VSEC structure 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)

This Register is only included if C_INCLUDE_BAR_OFFSET_REG = 1.

Table 19: **VSEC Capability Register 2**

Bit(s)	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. Hard-coded 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 20) provides a unique (within a given vendor) identifier for the layout and contents of the VSEC structure, as well as its revision and length.

This Register is only included if C_INCLUDE_BAR_OFFSET_REG = 1.

Table 20: **VSEC Header Register 2**

Bit(s)	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. Hard-coded to 0x0.
31:20	VSEC Length	RO	0x038	Length of the entire VSEC Capability structure, in bytes, including the VSEC Capability register. Hard-coded 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 21 and the register bits are described in Table 22. This set of registers can be used in two configurations based on the top-level parameter C_AXIBAR_AS_n. When the 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 BAR number. When the BAR is set to a 64-bit address space, then the translation's 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.

Table 21: **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 22: **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)	**Lower Address**: 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
				Upper Address: 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)	**Lower Address**: 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
				Upper Address: 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)	**Lower Address**: 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_2 = 64 bits) then reset value = C_AXIBAR2PCIEBAR_2(63 to 32)
			
			if (C_AXIBAR2PCIEBAR_2 = 32 bits) then reset value = 0x00000000
				Upper Address: 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)	**Lower Address**: 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
				Upper Address: 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)	**Lower Address**: To create the address for PCIe–this is the value substituted for the least significant 32 bits of the AXI address.

Table 22: **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_4 = 64 bits) then reset value = C_AXIBAR2PCIEBAR_4(63 to 32)
			
			if (C_AXIBAR2PCIEBAR_4 = 32 bits) then reset value = 0x00000000
				Upper Address: 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)	**Lower Address**: 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
				Upper Address: 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 PCIE is configured as a Root Complex, configuration traffic is generated by using the PCI Express Enhanced Configuration Access Mechanism (ECAM). ECAM functionality is available only if C_INCLUDE_RC = 1 and C_FAMILY = 7 series. Reads and writes to a certain memory aperture are translated to configuration reads and writes, as specified in the *PCI Express Base Specification (v1.1 and v2.1), §7.2.2*.

Depending on the core configuration, the ECAM memory aperture is 2^{21}–2^{28} (byte) addresses. The address breakdown is defined in Table 23. The ECAM window begins at memory map base address and extends to $2^{(20+ECAM\_SIZE)}$ - 1. ECAM_SIZE is calculated from the C_BASEADDR and C_HIGHADDR parameters. The number N of low-order bits of the two parameters that do not match, specifies the 2**n byte address range of the ECAM space. If C_INCLUDE_RC = 0, then ECAM_SIZE = 0.

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 Complex 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, the bridge does not generate a configuration request and signal SLVERR response on the AXI4-Lite bus. When the AXI Bridge for PCIE is configured for EP (C_INCLUDE_RC = 0), 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 PCIe is configured for RC (C_INCLUDE_RC = 1), the same is true, but it also looks like an ECAM access to primary bus, 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 23: **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 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.
(20+n-1):20	Bus Number	Bus Number 1 <= n <= 8. n is number of bits available for Bus Number as derived from core parameters C_INCLUDE_RC, C_BASEADDR and C_HIGHADDR.

Clock and Reset Interface

Clock Interface

The AXI_ACLK_OUT output must be fed back and used for the input of AXI_ACLK. This is the main memory mapped AXI4 bus clock.

The REFCLK input must be provided at the frequency selected by the value of C_REF_CLK_FREQ. This clock is used to generate the two output clocks and is also the clock used to drive the AXI4 bus.

The AXI_CTL_ACLK_OUT output must be fed back and used for the input AXI_CTL_ACLK. This is the AXI4-Lite interconnect clock. The AXI_CTL_ACLK_OUT clock is rising edge aligned and an integer division of the AXI_ACLK_OUT clock.

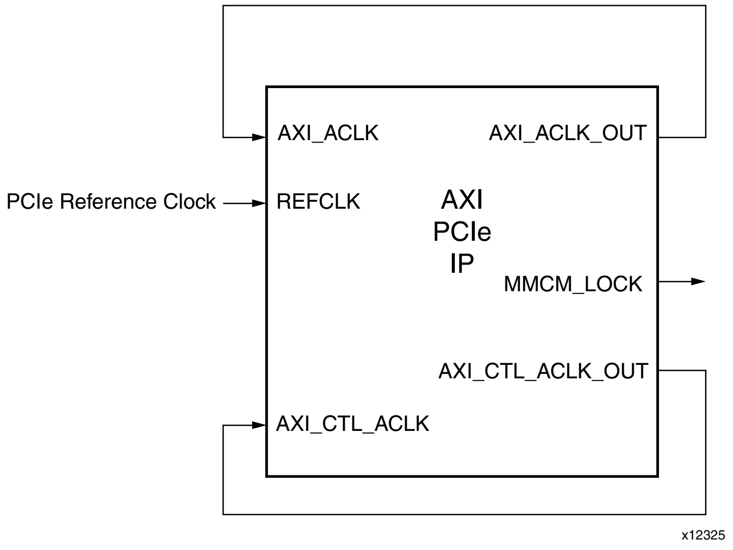

Figure 2: **Clocking Diagram**

Reset Interface

The bridge is designed to be used with the Proc_Sys_Reset module for generation of the `AXI_ARESET` input. When using the Embedded Development Kit (EDK) tools to build a system, it is best to connect the `PERSTN` pin of the host connector for PCIe to the `Peripheral_Reset` port of the Proc_Sys_Reset module. The bridge does not use `PERSTN` directly. Also, the `MMCM_LOCK` output must be connected to the `DCM_Locked` input of the Proc_Sys_Reset module to make sure that `AXI_ARESET` is held active for 16 clocks after `MMCM_LOCK` becomes active. See Figure 3.

Note: Be sure to set the correct polarity on the `Aux_Reset_In` signal of the Proc_Sys_Reset IP block. When `PERSTN` is active low, set the parameter as follows.

```
    PARAMETER C_AUX_RESET_HIGH = 0
```

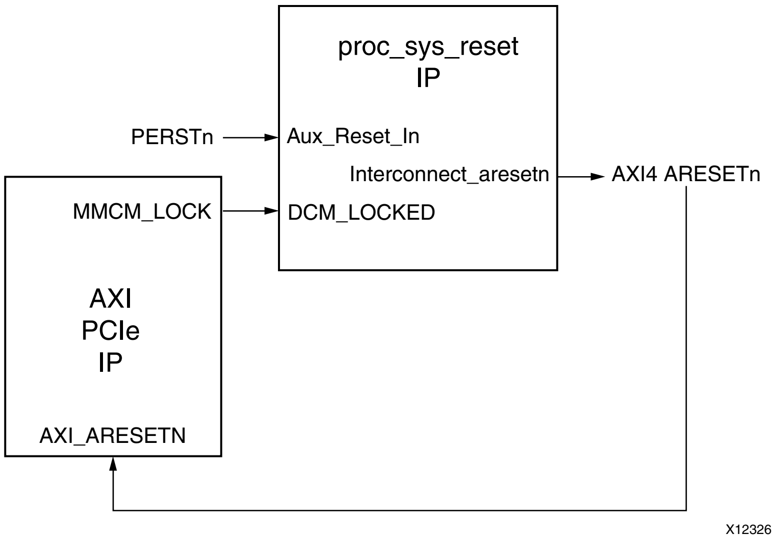

Figure 3: **System Reset Connection**

AXI Transactions for PCIe

Table 24 andTable 25 are the translations tables for AXI4-Stream and memory mapped transactions

Table 24: **AXI4 Memory Mapped Transactions to AXI4-Stream PCIe TLPs**

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

Table 25: **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 with 32-bit address
MemWr 32 (3DW) to PCIEBAR	INCR Burst Write with 32-bit address
MemRd 64 (4DW) of PCIEBAR	INCR Burst Read with 32-bit address
MemWr 64 (4DW) to PCIEBAR	INCR Burst Write with 32-bit address

Transaction Ordering for PCIe

The AXI Bridge for PCIe conforms to strict PCIe transaction ordering rules. See the PCIe v2.1 Specification for the complete rule set. The following behaviors are implemented in the AXI Bridge for PCIe 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 TPL 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 AXI master read of a remote PCIe device is not permitted to pass any previous or simultaneous AXI master writes to a remote PCIe device that occurs previously or at the same time. Timing is based off the AXI ARVALID signal timing relative to the AXI AWVALID. Any AXI write transaction in which AWVALID was asserted before or at the same time as the ARVALID for a read from pcie is asserted causes the MemRd TLP(s) to be held until the pipelined or simultaneous MemWr TLP(s) have been sent.
- 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 PCIe. 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 PCIe 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.
- Read data from a remote AXI slave is not permitted to pass any remote AXI master writes to a remote PCIe device initiated on the AXI bus prior to or simultaneously with the read data being returned on the AXI bus. Timing is based off the AXI AWVALID signal timing relative to the AXI RVALID assertion. Any AXI write transaction in which AWVALID was asserted before or simultaneously with the RVALID being asserted up to and including the last data beat, causes the Completion TLP(s) to be held until the pipelined or simultaneous MemWr TLP(s) have been sent.

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

Address Translation

The address space for PCIe is different than 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, C_AXIBAR2PCIEBAR_n and C_AXIBAR_AS_n,

where "n" represents an AXI BAR number from 0 to 5. The bridge for PCIe supports mapping on up to three 64-bit BARs for PCIe. The generics used to configure the BARs are:

C_PCIEBAR_NUM, C_PCIE2AXIBAR_n and C_PCIEBAR_LEN_n,

where "n" represents a particular BAR number for PCIe from 0 to 2.
The C_INCLUDE_BAROFFSET_REG generic allows for dynamic address translation. When this parameter is set to one, the AXIBAR2PCIEBAR_n translation vectors can be changed by using the software.

In the four following examples,

- Example 1 (32-bit PCIe Address Mapping) demonstrates how to set up four 32-bit AXI BARs and translate the AXI address to an address for PCIe.
- Example 2 (64-bit PCIe Address Mapping) demonstrates how to set up three 64-bit AXI BARs and translate the AXI address to an 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 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 four independent 32-bit AXI BARs and address translation of AXI addresses to a remote address space for PCIe. This setting of AXI BARs does not depend on the BARs for PCIe within the AXI Bridge for PCIe.

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

```
C_AXIBAR_AS_0=0
C_AXIBAR_0=0x12340000
C_AXI_HIGHADDR_0=0x1234FFFF
C_AXIBAR2PCIEBAR_0=0x5671XXXX (Bits 15-0 are don't cares as the lower 16-bits will hold the
actual lower 16-bits of the PCIe address)

C_AXIBAR_AS_1=0
C_AXIBAR_1=0xABCDE000
C_AXI_HIGHADDR_1=0xABCDFFFF
C_AXIBAR2PCIEBAR_1=0xFEDC0XXX (Bits 12-0 are don't cares as the lower 13-bits will hold the
actual lower 13-bits of the PCIe address)

C_AXIBAR_AS_2=0
C_AXIBAR_2=0xFE000000
C_AXI_HIGHADDR_2=0xFFFFFFFF
C_AXIBAR2PCIEBAR_2=0x40XXXXXX (Bits 24-0 are don't cares)
```

- Accessing the Bridge AXIBAR_0 with address `0x12340ABC` on the AXI bus yields `0x56710ABC` on the bus for PCIe.

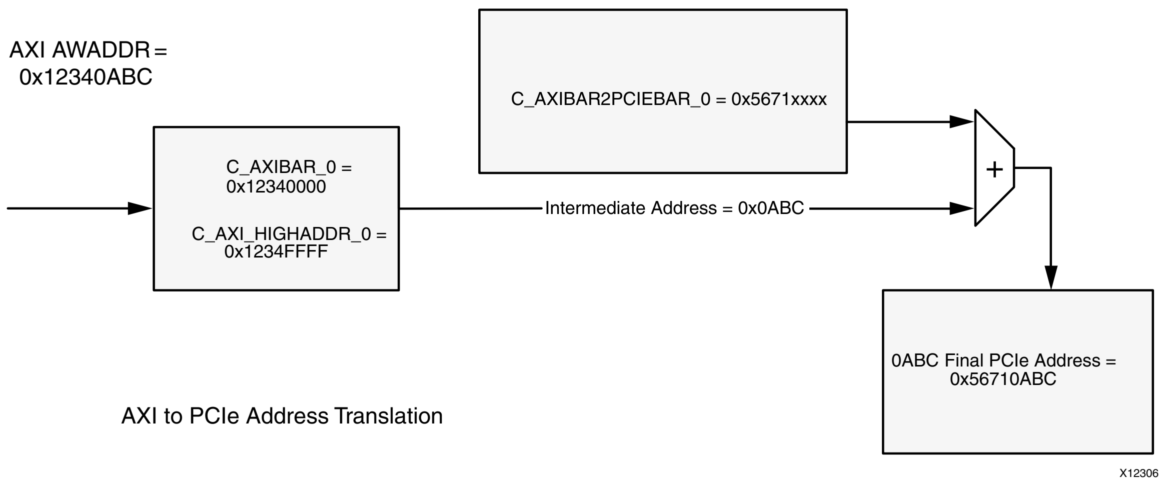
AXI AWADDR = 0x12340ABC
C_AXIBAR_0 = 0x12340000
C_AXI_HIGHADDR_0 = 0x1234FFFF
C_AXIBAR2PCIEBAR_0 = 0x5671xxxx
Intermediate Address = 0x0ABC
+
0ABC Final PCIe Address = 0x56710ABC
AXI to PCIe Address Translation
X12306

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

- Accessing the Bridge AXIBAR_1 with address `0xABCDF123` on the AXI bus yields `0xFEDC1123` on the bus for PCIe.
- Accessing the Bridge AXIBAR_2 with address `0xFFFEDCBA` 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 64-bit AXI BARs and address translation of AXI addresses to a remote 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:

```
C_AXIBAR_AS_0=1
C_AXIBAR_0=0x12340000
C_AXI_HIGHADDR_0=0x1234FFFF
C_AXIBAR2PCIEBAR_0=0x500000005671XXXX (Bits 15-0 are don't cares)

C_AXIBAR_AS_1=1
C_AXIBAR_1=0xABCDE000
C_AXI_HIGHADDR_1=0xABCDFFFF
C_AXIBAR2PCIEBAR_1=0x60000000FEDC0XXX (Bits 12-0 are don't cares)

C_AXIBAR_AS_2=1
C_AXIBAR_2=0xFE000000
C_AXI_HIGHADDR_2=0xFFFFFFFF
C_AXIBAR2PCIEBAR_2=0x7000000040XXXXXX (Bits 24-0 are don't cares)
```

- Accessing the Bridge AXIBAR_0 with address `0x12340ABC` on the AXI bus yields `0x5000000056710ABC` on the bus for PCIe.
- Accessing the Bridge AXIBAR_1 with address `0xABCDF123` on the AXI bus yields `0x60000000FEDC1123` on the bus for PCIe.
- Accessing the Bridge AXIBAR_2 with address `0xFFFEDCBA` 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:

```
BAR 0 is set to 0x20000000_ABCD8000 by the Root Complex
C_PCIEBAR_LEN_0=15
C_PCIEBAR2AXIBAR_0=0x1234_0XXX (Bits 14-0 are don’t cares)

BAR 1 is set to 0xA000000012000000 by Root Complex
C_PCIEBAR_LEN_1=25
C_PCIEBAR2AXIBAR_1=0xFEXXXXXX (Bits 24-0 are don’t cares)
```

- Accessing the Bridge PCIEBAR_0 with address `0x20000000_ABCDFFF4` on the bus for PCIe yields `0x1234_7FF4` on the AXI bus.

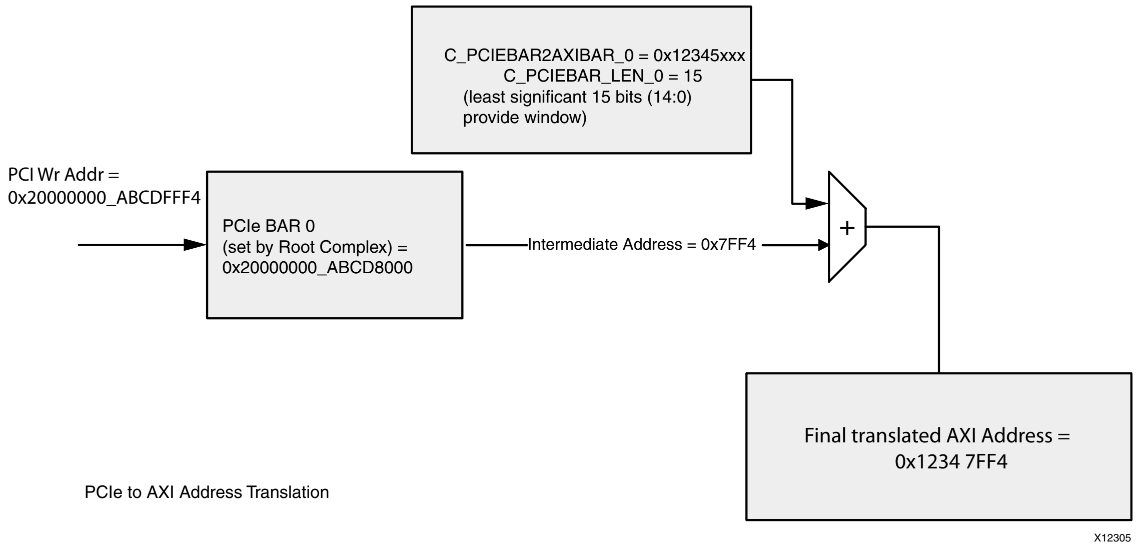
C_PCIEBAR2AXIBAR_0 = 0x12345xxx
C_PCIEBAR_LEN_0 = 15
(least significant 15 bits (14:0) provide window)
PCI Wr Addr = 0x20000000_ABCDFFF4
PCIe BAR 0 (set by Root Complex) = 0x20000000_ABCD8000
Intermediate Address = 0x7FF4
+
Final translated AXI Address = 0x1234 7FF4
PCIe to AXI Address Translation
X12305

Figure 5: **PCIe to AXI Translation**

- Accessing Bridge PCIEBAR_1 with address `0xA00000001235FEDC` on the bus for PCIe yields `0xFE35FEDC` on the AXI bus.

Example 4

This example shows the generic settings to set up a combination of two independent 32-bit AXI BARs and two independent 64-bit BARs and address translation of AXI addresses to a remote 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=4, the following assignments for each range are made:

```
C_AXIBAR_AS_0=0
C_AXIBAR_0=0x12340000
C_AXI_HIGHADDR_0=0x1234FFFF
C_AXIBAR2PCIEBAR_0=0x5671XXXX (Bits 15-0 are don’t cares)

C_AXIBAR_AS_1=1
C_AXIBAR_1=0xABCDE000
C_AXI_HIGHADDR_1=0xABCDFFFF
C_AXIBAR2PCIEBAR_1=0x50000000FEDC0XXX (Bits 12-0 are don’t cares)

C_AXIBAR_AS_2=0
C_AXIBAR_2=0xFE000000
C_AXI_HIGHADDR_2=0xFFFFFFFF
C_AXIBAR2PCIEBAR_2=0x40XXXXXX (Bits 24-0 are don’t cares)

C_AXIBAR_AS_3=1
C_AXIBAR_3=0x00000000
C_AXI_HIGHADDR_3=0x0000007F
C_AXIBAR2PCIEBAR_3=0x600000008765438X (Bits 6-0 are don’t cares)
```

- Accessing the Bridge AXIBAR_0 with address `0x12340ABC` on the AXI bus yields `0x56710ABC` on the bus for PCIe.
- Accessing the Bridge AXIBAR_1 with address `0xABCDF123` on the AXI bus yields `0x50000000FEDC1123` on the bus for PCIe.
- Accessing the Bridge AXIBAR_2 with address `0xFFFEDCBA` on the AXI bus yields `0x41FEDCBA` on the bus for PCIe.
- Accessing the AXI M S PCIe Bridge AXIBAR_3 with address `0x00000071` on the AXI bus yields `0x600000008765 43F1` on the bus for PCIe.

Addressing Checks

When setting the following parameters for PCIe address mapping, C_PCIE2AXIBAR_n and C_PCIEBAR_LEN_n, be sure these are set to allow for the 32-bit addressing space on the AXI system. For example, the following setting is illegal and results in an invalid AXI address.

```
C_PCIE2AXIBAR_0 = 0xFFFF_0000
C_PCIEBAR_LEN_0 = 23
```

Also, check for a larger value on C_PCIEBAR_LEN_n compared to the value assigned to parameter, C_PCIE2AXIBAR_n. For example, the following parameter settings.

```
C_PCIE2AXIBAR_0 = 0xFFFF_E000
C_PCIEBAR_LEN_0 = 20
```

To keep the AXI BAR upper address bits as 0xFFFF_E000 (to reference bits [31:13]), the C_PCIEBAR_LEN_0 parameter must be set to 13.

Interrupts

This section describes the interrupt pins which include Local, MSI and Legacy Interrupts.

Local Interrupts

The `INTERRUPT_OUT` pin can be configured to send interrupts based on the settings of the Interrupt Mask register. The `INTERRUPT_OUT` pin signals interrupts to devices attached to the memory mapped AXI4 side of the Bridge. The MSI interrupt defined in the Interrupt Mask & Interrupt Decode registers is used to indicate the receipt of a Message Signaled Interrupt only when the bridge is operating in Root Complex mode (C_INCLUDE_RC=1).

MSI Interrupt

When the `MSI_enable` output pin indicates that the bridge has Endpoint MSI functionality enabled (`MSI_enable` = '1'), the `INTX_MSI_Request` input pin is defined as MSI_Request and can be used to trigger a Message Signaled Interrupt via a special MemWr TLP to an external Root Complex for PCIe on the PCIe side of the Bridge. The `INTX_MSI_Request` input pin is positive-edge detected and synchronous to `AXI_ACLK`. The address and data contained in this MemWr TLP are determined by an external Root Complex for PCIe configuration of registers within the integrated block for PCI Express. The `INTX_MSI_Request` pin input is valid only when the bridge is operating in Endpoint mode (C_INCLUDE_RC=0).

Legacy Interrupts

The bridge supports legacy interrupts for PCI® if selected by the C_INTERRUPT_PIN parameter. (Can only be set to 1 when C_INCLUDE_RC = 0). A value of 1 selects INTA, as defined in Table 1. If a legacy interrupt for PCI support is selected and the `MSI_enable` output pin indicates that the bridge has end point MSI functionality disabled (`MSI_enable` = '0'), the `INTX_MSI_Request` pin is defined as `INTX`. When the `INTX` pin goes high, an assert `INTA` message is sent. When the `INTX` pin goes low, a deassert `INTA` message is sent. These messages are defined in the PCI 2.1 specification. The `INTX_MSI_Request` pin input is valid only when the bridge is operating in Endpoint mode (C_INCLUDE_RC=0).

Malformed TLP

The integrated PCIe block in the FPGA for PCIe Express will detect 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.

For the IP configured as a Root Port, when a Malformed TLP is received from the Endpoint, this may fall under one of several types of violations as per the PCIe specification. For example, if a Received TLP has the Error Poison bit set, this is discarded by the MM/S master bridge, and the MEP (Master Error Poison) bit is set in the Interrupt Decode Register.

Abnormal Conditions

This section describes how the Slave side (Table 26) and Master side (Table 27) of the AXI Bridge for PCI Express 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 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 may 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 may 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 `SLVERR` 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 26: **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. SLVERR 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 adhere 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 corrupt. 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 27: **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
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 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 PCIe, 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 will have a `SLVERR` response given and the write data will be discarded. Initiated AXI4 read transactions that have not yet completed on the AXI4 bus when the link goes down will 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 will complete as normal. Any MemRd TLPs for PCIe that have been received, but have not returned completion TLPs by the time the link goes down, will complete on the AXI4 bus, but will not return completion TLPs on the PCIe bus.

Root Complex

When configured to support Root Complex functionality (only available with 7 series FPGA implementations), the AXI Bridge for PCIe fully supports Root Complex operation as supported by the underlying block. There are a few details that need special consideration. The following subsections contain important information and design considerations about Root Complex support.

Power Limit Message TLP

The AXI Bridge for PCIe 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 Complex software to the type 1 PCI Configuration Header of the AXI Bridge for PCIe 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 PCIe IP core will assert `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.

If a configuration transaction is attempted to a device number other than zero, the AXI Bridge for PCIe asserts `SLVERR` on the AXI4-Lite bus. PCIe transactions are generated for only the device number of zero.

Configuration Transaction Timeout

Configuration transactions are non-posted transactions. The AXI Bridge for PCIe IP 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. Timeout 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 28.

Table 28: **Responses of AXI Bridge for PCIe 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	Device number not zero	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

Line Rate Support for PCIe Gen1/Gen2

The link speed, number of lanes supported, and support of line rate for PCIe are defined in Table 29. Achieving line rate for PCIe is dependent on the device family, the AXI clock frequency, the AXI data width, the number of lanes, and Gen 1 or Gen 2 link speed.

Table 29: **Line Rate for PCIe Support for Gen1/Gen2**

C_FAMILY	AXI_ACLK Frequency	C_X_AXI_DATA_WIDTH	C_NO_OF_LANES	Gen 1 (2.5 GT/s)	Gen 2 (5.0 GT/s)
Spartan-6	62.5 MHz	32	x1	Yes	No
Virtex-6, Kintex-7	125 MHz	64	x1	Yes	Yes
Virtex-6, Kintex-7	125 MHz	64	x2	Yes	Yes
Virtex-6, Kintex-7	125 MHz	64	x4	Yes	No
Virtex-6	125 MHz	64	x8	No	No
Kintex-7	125 MHz	128	x4	Yes	Yes
Kintex-7	125 MHz	128	x8	Yes	No

Limitations

Reference Clock for PCIe Frequency Value

The REFCLK input used by the serial transceiver for PCIe must be 100 MHz or 125 MHz for Spartan-6 device configurations, and 100 MHz or 250 MHz for Virtex-6 and 7 series device configurations. The C_REF_CLK_FREQ parameter is used to set this value, as defined in Table 1.

AXI Clock Frequency Values

- 62.5 MHz for Spartan-6 FPGA devices
- 125 MHz for Virtex-6 and 7-Series FPGA devices

Constraints

The AXI Bridge for PCIe IP core requires a clock period constraint for the REFCLK input that agrees with the C_REF_CLK_FREQ parameter setting. In addition, pin-placement (LOC) constraints are needed that are board/part/package specific.

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

Additional information is available in the Answer Record support on www.xilinx.com.

Device Utilization and Performance Benchmarks

System Performance

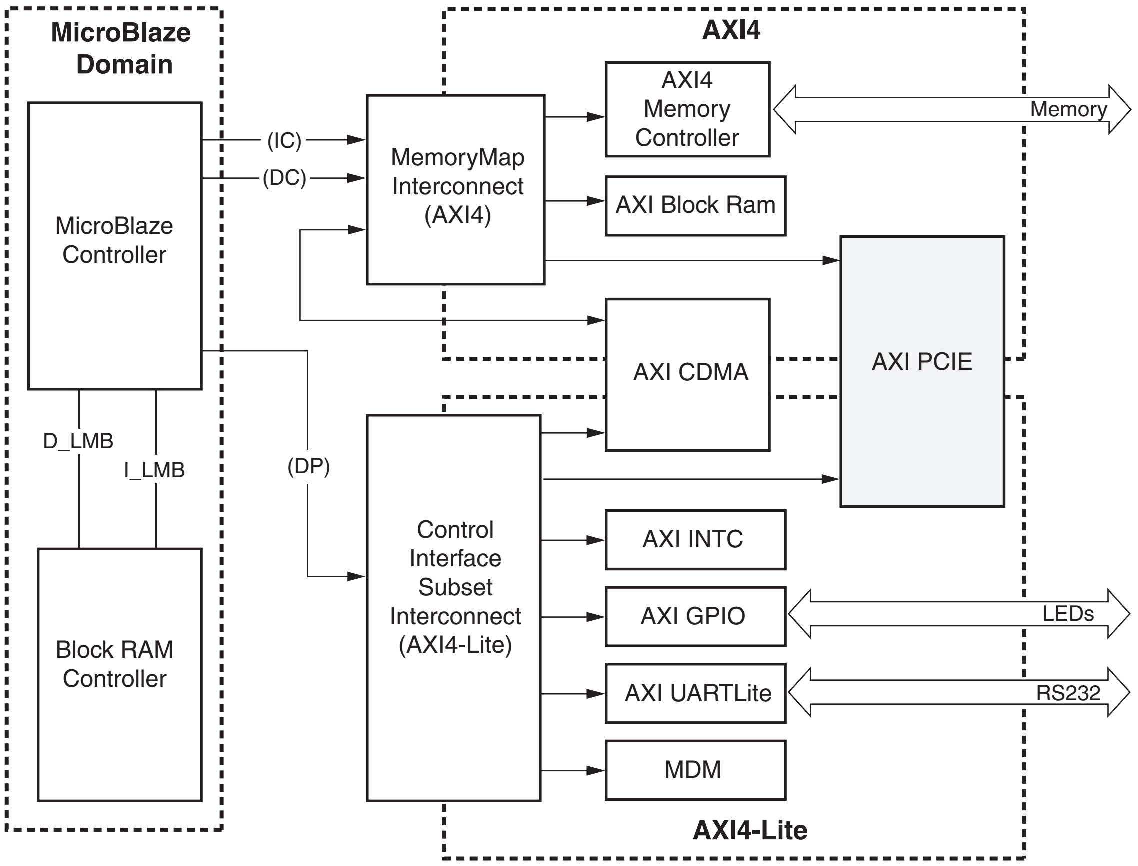

Figure 6: **FPGA System Configuration Diagram**

The target FPGA is filled with logic to drive the Lookup Table (LUT) and block Random Access Memory (RAM) utilization to approximately 70% and the Input/Output (I/O) utilization to approximately 80%. The data shown in Table 30 is obtained using the default tool options and the slowest speed grade for the target FPGA.

Table 30: **System Performance**

Target FPGA	Target F_{MAX} (MHz)		
	AXI4	**AXI4-Lite**	**MicroBlaze**
XC6SLX45T [1]	90 MHz	120 MHz	80 MHz
XC6VLX240T [2]	135 MHz	180 MHz	135 MHz

Notes:

1. Spartan-6 FPGA LUT utilization: 70%; block RAM utilization: 70%; I/O utilization: 80%; MicroBlaze processor not AXI4 interconnect; AXI4 interconnect configured with a single clock of 120 MHz.
2. Virtex-6 FPGA LUT utilization: 70%; block RAM utilization: 70%; I/O utilization: 80%.
3. Kintex-7 FPGA results are not shown, but expected to be comparable to Virtex-6 devices.

System Integration

A typical embedded system including the AXI Bridge for PCIe is shown in Figure 6. Some additional components to this system in the EDK environment can include the need to connect the MicroBlaze™ processor peripheral ports to communicate with PCI Express (in addition to the AXI4-Lite register port on the PCIe bridge). The EDK provides a helper core to achieve this functionality and bridges transactions from the AXI4-Lite MicroBlaze processor peripheral ports (DP and IP) to the AXI4 Interconnect (connected to the AXI Bridge for PCIe). The axi2axi_connector IP core provides this support and you can connect this in the EDK environment.

The AXI Bridge for PCIe IP core can be configured with each port connection for an AXI EDK 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 EDK system may need to use a differential input buffer (that is instantiated separately) from the AXI Bridge for the PCIe IP core. This can be accomplished with a separate IP block available in the EDK under the name util_ds_buf.

The I/O buffer instantiation for the PCIe differential clock on Spartan-6 FPGA design implementations is described here. The buffer type is specific for Spartan-6 FPGAs.

```
BEGIN util_ds_buf
 PARAMETER INSTANCE = PCIe_Diff_Clk_I
 PARAMETER HW_VER = 1.01.a
 PARAMETER C_BUF_TYPE = IBUFDS
 PORT IBUF_DS_P = PCIe_Diff_Clk_P
 PORT IBUF_DS_N = PCIe_Diff_Clk_N
 PORT IBUF_OUT = PCIe_Diff_Clk
END
```

A Virtex-6 FPGA I/O buffer instantiation follows (as in the MHS file). The buffer type is specific for the Virtex-6 architecture.

```
BEGIN util_ds_buf
 PARAMETER INSTANCE = PCIe_Diff_Clk_I
 PARAMETER HW_VER = 1.01.a
 PARAMETER C_BUF_TYPE = IBUFDSGTXE
 PORT IBUF_DS_P = PCIe_Diff_Clk_P
 PORT IBUF_DS_N = PCIe_Diff_Clk_N
 PORT IBUF_OUT = PCIe_Diff_Clk
END
```

A Kintex-7 FPGA I/O buffer instantiation follows (as it would appear in the MHS file). The buffer type that follows is specific to Kintex-7 FPGA implementations. The buffer type can change based on target FPGA technology.

```
BEGIN util_ds_buf
 PARAMETER INSTANCE = PCIe_Diff_Clk_I
 PARAMETER HW_VER = 1.01.a
 PARAMETER C_BUF_TYPE = IBUFDSGTE
 PORT IBUF_DS_P = PCIe_Input_Clk_P
 PORT IBUF_DS_N = PCIe_Input_Clk_N
 PORT IBUF_OUT = PCIe_Diff_Clk
END
```

The corresponding location constraints for the PCIe clock differential buffer are highlighted in the following section.

Placement Constraints

For designers utilizing the Base System Builder in the EDK, placement and timing constraints are auto-generated for specific FPGA technologies. These include the SP605, ML605 and KC705. It is recommended to use the BSB tool as a guide for creating the necessary placement and timing constraints on the AXI Bridge for PCIe IP core.

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

- the PCIe embedded block itself,
- the PCIe GTX transceivers (for each channel), and
- the PCIe differential clock input (if utilized)

Because these blocks are embedded in the AXI Bridge for PCIe bridge IP core, the following path constraints must be utilized. The path names for each of these blocks vary based on the FPGA architecture. The following sections describe example constraints for each supported FPGA architecture: Spartan-6, Virtex-6, and Kintex-7. See the sections, Constraints for Spartan-6 Devices, Constraints for Virtex-6 Devices, and Constraints for Kintex-7 FPGAs). All example constraints provided in this document are supported in the UCF for EDK designs.

Constraints for Spartan-6 Devices

For placement constraints on Spartan-6 FPGA designs, see the *Spartan-6 GTP Transceiver User Guide* for more information. Spartan-6 FPGA designs can LOC the I/Os on the PCIe design to use the corresponding serial transceivers in the UCF.

Constraints for Virtex-6 Devices

This section highlights the LOC constraints to be specified in the UCF for the AXI Bridge for PCIe IP core for Virtex-6 FPGA design implementations. The "*" characters must be specified for the FPGA/board design.

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

```
INST "*/pcie_2_0_i/pcie_block_i" LOC = PCIE_X*Y*;
```

For placement/path information of the PCIe GTX transceivers, the following constraint can be utilized.

```
INST "*/pcie_2_0_i/pcie_gt_i/gtx_v6_i/GTXD[0].GTX" LOC = GTXE1_X*Y*;
```

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

```
INST "*/PCIe_Diff_Clk_I/USE_IBUFDS_GTXE1.GEN_IBUFDS_GTXE1[0].IBUFDS_GTXE1_I" LOC =
IBUFDS_GTXE1_X*Y*;
```

The "`PCIe_Diff_Clk_I`" is the name of the instance (in the MHS file) given to the differential buffer (util_ds_buf) in the MHS file as highlighted in the section, System Integration.

Constraints for Kintex-7 FPGAs

This section highlights the LOC constraints to be specified in the UCF for the AXI Bridge for PCIe IP core for Kintex-7 FPGA design implementations. The "*" characters must be specified for the FPGA/board design.

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

```
INST "*pcie_7x_v1_3*/*pcie_top_i/pcie_7x_i/pcie_block_i" LOC = PCIE_X*Y*;
```

For placement/path information of the PCIe GTX transceivers, the following constraint can be utilized.

```
INST
"*pcie_7x_v1_3*/*gt_top_i/pipe_wrapper_i/pipe_lane[*].gt_wrapper_i/gtx_channel.gtxe2_chann
el_i" LOC = GTXE2_CHANNEL_X*Y*;
```

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

```
INST "*/PCIe_Diff_Clk_I/USE_IBUFDS_GTE2.GEN_IBUFDS_GTE2[0].IBUFDS_GTE2_I" LOC =
IBUFDS_GTE2_X*Y*;
```

The "`PCIe_Diff_Clk_I`" is the name of the instance (in the MHS file) given to the differential buffer (util_ds_buf) in the MHS file as highlighted in the section, System Integration, page 41.

Kintex-7 FPGA Root Port (x2 Gen1) Device Utilization

Table 31: **Kintex-7 FPGA Root Port (x2 Gen1)**

Slices	Registers	LUTs
4700	8200	10800

Kintex-7 FPGA Endpoint (x2 Gen1) Device Utilization

Table 32: **Kintex-7 FPGA Endpoint (x2 Gen1)**

Slices	Registers	LUTs
3600	6300	9050

There is additional design parameter for 7 series FPGA implementations, C_PCIE_USE_MODE. This parameter is introduced with EDK 13.4 and the AXI Bridge for PCIe v1.02.a release. Set the parameter accordingly as noted in Table 1 for IES and GES silicon desired implementations.

Specification Exceptions

N/A

Reference Documents

To search for Xilinx documentation, go to www.xilinx.com/support/documentation/index.htm

1. *AMBA AXI Protocol Specification*
2. Xilinx AXI4 Memory-Mapped Protocol Specification Version 1.7
3. AXI4-Stream PCIe Bridge Design Specification
4. PCI Express Base Specification, v1.1 (§7.19 of v2.0)
5. PCI Express Base Specification (v1.1 and v2.0), §7.2.2
6. PCI Express Base Specification Revision 2.1

Support

Xilinx provides technical support for this LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled *DO NOT MODIFY*.

Ordering and Licensing Information

This Xilinx® LogiCORE IP module is provided at no additional cost with the Xilinx ISE® Design Suite Embedded Edition software under the terms of the Xilinx End User License. The core is generated using the Xilinx ISE Embedded Edition software (EDK).

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

Revision History

Date	Version	Revision
6/22/11	1.0	Initial Xilinx Release.
9/13/11	1.1	Release with EDK 13.3.
10/19/11	1.2	Summary of Documentation Changes • Updated Bridge information and PHY status/control register per AXI-Enhanced Architecture Specification v0.9. • Added all C_INTERCONNECT_* parameters to the end of Table 1 • Added description about use of axi2axi_connector IP core in AXI Bridge for PCIe systems. Added a note that inclusion of this helper core is for MicroBlaze and XMD access to the AXI Bridge for PCIe • Added Figures 4 and to pages 25 and 27 • Added System Integration section on page 33
01/18/12	1.3	Summary of Major Documentation Changes • Updated register mapping. • Removed List of Acronyms. • Added the following sections: • Root Port Status/Control Register (Offset 0x148) • Root Port MSI Base Register 1 (Offset 0x14C) • Root Port MSI Base Register 2 (Offset 0x150) • Root Port Error FIFO Read Register (Offset 0x154) • Root Port Interrupt FIFO Read Register 1 (Offset 0x158) • Root Port Interrupt FIFO Read Register 2 (Offset 0x15C) • Root Complex • Added Addressing Checks, under Address Translation. • Added Kintex-7 FPGA Root Port (x2 Gen1) Device Utilization and Kintex-7 FPGA Endpoint (x2 Gen1) Device Utilization, • Added C_PCIE_USE_MODE parameter. • Added Placement Constraints section. • Updated all reference of "AXI PCIe" to "AXI Bridge for PCIe" • Updated Figure 5.

Notice of Disclaimer