LogiCORE IP PLBv46 RC/EP Bridge for PCI Express (v4.06.a)

DS540 December 14, 2010 **Product Specification**

Introduction

This document defines the functional operation of the PLBv46 Root Complex and Endpoint Bridge for PCI Express®, hereafter called PLBv46 Bridge. The PLBv46 Bridge is an interface between the Processor Local Bus (PLB) and the PCI Express (PCIe®) bus.

This document provides the definitions for all of the functional modules, registers, and interfaces that need to be implemented in the PLBv46 Bridge. This document defines the hardware implementation and software interfaces to the PLBv46 Bridge in a FPGA.

Features

- Configurable Root Complex or Endpoint functionality for Virtex®-6 FPGA
- Endpoint functionality for Virtex-5 and Spartan®-6 FPGAs
- Supports PLB access to PCIe space
- Supports PCIe access to PLB space
- Translates PLB transactions to appropriate PCIe Transaction Layer Packets (TLPs)
- Tracks and manages TLPs that require completion processing
- Indicates error conditions detected by the PCIe core through interrupts
- Supports up to six 32-bit or six 64-bit remote PLB Base Address Register (BAR) regions mapped to PCIe address space
- Address spaces are defined with a base address, an upper address, space type (I/O or Memory) and an address translation value
- I/O space with 32-bit address supported when the Virtex-6 FPGA is configured as Root Complex
- Supports up to three 32-bit or 64-bit PCIe Base Address Register (BAR) memory regions mapped to PLB address space when configured as Endpoint

LogiCORE™ IP Facts		
Core Specifics		
Supported Device Family[1]	Virtex-6, Spartan-6, Virtex-5	
Version of core	plbv46_pcie	v4.06.a
Resources Used	See Table 19.	
Provided with Core		
Documentation	Product Specification	
Design File Formats	VHDL	
Constraints File	.ucf (user constraints file)	
Verification	VHDL Test Bench	
Instantiation Template	VHDL Wrapper	
Design Tool Requirements		
Xilinx Implementation Tools	ISE v12.4 software	
Verification	Mentor Graphics ModelSim 6.5c	
Simulation	Mentor Graphics ModelSim 6.5c	
Synthesis	XST	
Support		
Provided by Xilinx, Inc.		

[1] For a complete listing of supported devices, see the release notes for this core.

Features (continued)

- Supports one 32-bit or 64-bit PCIe Base Address Register (BAR) memory region mapped to PLB address space when the Virtex-6 FPGA is configured as Root Complex
- Independent PLB and PCIe clocks
- Supports 32/64-bit PLB version 4.6
- Supports Spartan-6 FPGA x1 PCIe lane configuration at 2.5 GigaTransfers per second (GT/s)
- Supports Virtex-5 FPGA x1, x4, and x8 PCIe lane configuration at 2.5GT/s
- Supports Virtex-6 FPGA x1 PCIe lane configuration at 2.5 GT/s
- Full bridge functionality
  - PLB master read and write of a remote PCIe target (both single and burst)
  - PCIe requester read and write to a remote PLB slave (both single and multiple)

Functional Description

The PLBv46 Bridge provides transaction level translation of PLB bus commands to PCIe TLP packets and PCIe requests to PLB bus commands. The architecture of the PLBv46 Bridge is shown in Figure 1. The PLBv46 Bridge is composed of seven core sections: Slave IPIF, Master IPIF, Management/Register Block, Slave Bridge, Master Bridge, and Transaction Layer Interface (TLIF) arbiter and the Core Wrapper.

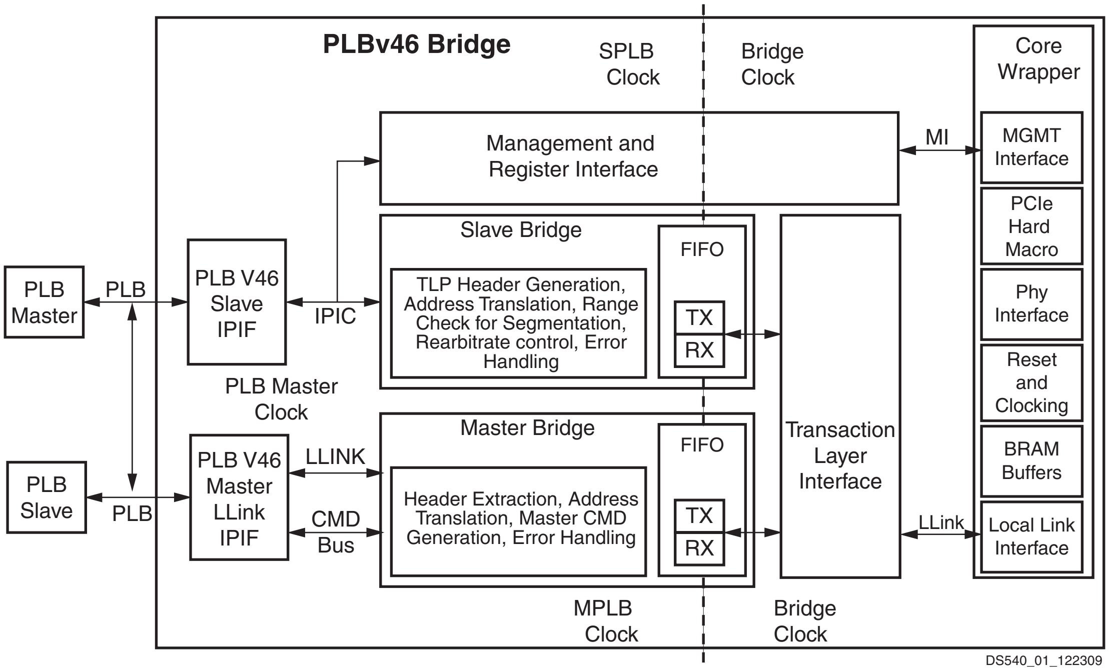

Figure 1: **PLBv46 Bridge Architecture**

The Slave IPIF provides termination of PLB transactions to the Bridge from a PLB master device such as a Processor. The Bridge provides a means to translate addresses that are mapped within the PLB address domain to PCIe domain addresses. Commands from the PLB master are converted into a TLP request and queued in the Slave Bridge TX FIFO. Packets in this queue are sent to the PCIe core via the TLIF arbiter. The TLIF arbiter round robins the two TX queues (one for PLB requests and the other for PCIe completions) to determine which one will access the single interface on the hard PCIe core.

For PLB read requests, the TLP will be written into the PLB Bridge TX FIFO and await transmission to the PCIe hard core. Once the TLIF arbiter sends this read request to the PCIe core the TLIF arbiter will begin the completion time-out counter. The Slave Bridge will only accept one PLB read request at a time however it may accept as many PLB write requests as it can fit into the TX FIFO. Once the FIFO is full, further PLB commands will be rearbitrated.

The Management/Register Block houses the bulk of the registers used in the bridge. This includes the BARs, Interrupts, PCIe hard core registers, and miscellaneous status from the PCIe hard core.

The Master Bridge processes read and write command TLPs received from the TLIF and Core Wrapper and creates the appropriate PLB commands and manages the flow of data associated with each command between the PLB Master IPIF and the Core Wrapper.

PLBv46 Bridge Parameters

Because many features in the PLBv46 Bridge design can be parameterized, the user can realize a PLB to PCIe bridge uniquely tailored while 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 PLBv46 Bridge 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	virtex5, virtex6, spartan6	virtex5	String
G2	C_SUBFAMILY	“lx” - Selects V5LXT device. ““fx” - Selects V5FXT device Note: No effect for virtex6 and spartan6.	“LX”, “FX”	“lx”	String
G3	C_INCLUDE_RC	Configures the Bridge to be a Root Complex or an Endpoint	0 = Endpoint 1 = Root Complex	0	Integer
G4	C_BASEADDR	Device base address	Valid PLB address (1)(3)	0xFFFF_FFFF	std_logic_vector
G5	C_HIGHADDR	Device absolute high address	Valid PLB address (1)(3)	0x0000_0000	std_logic_vector
G6	C_ECAM_BASEADDR	ECAM base address	Valid PLB address (1)(3)(7)	0xFFFF_FFFF	std_logic_vector
G7	C_ECAM_HIGHADDR	ECAM high address	Valid PLB address (1)(3)(7)	0x0000_0000	std_logic_vector
G8	C_COMP_TIMEOUT	Selects the completion timeout counter value for PLB to PCIe non-posted transactions	0 = 50 uS 1 = 50 mS	0	Integer
G9	C_INCLUDE_ BAROFFSET_REG	Include the registers for high-order bits to be substituted in translation	0 = exclude 1 = include	0	Integer

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

Generic	Parameter Name	Description	Allowable Values	Default Value	VHDL Type
G10	C_IPIFBAR_NUM	Number of PLB 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, BAR_1, BAR2, BAR_3 enabled 5 = BAR_0, BAR_1, BAR_2, BAR_3, BAR_4 enabled 6 = BAR_0 through BAR_5 enabled	6	Integer
G11	C_IPIFBAR_0	PLB BAR_0 aperture low address	Valid PLB address (1)(3)(4) (5)	0xFFFF_FFFF	std_logic_vector
G12	C_IPIFBAR_HIGHADDR_0	PLB BAR_0 aperture high address	Valid PLB address (1)(3)(4)	0x0000_0000	std_logic_vector
G13	C_IPIFBAR_AS_0	PLB BAR_0 address size	0 = 32 bits 1 = 64 bits	0	Integer
G14	C_IPIFBAR_SPACE_TYPE_0	PLB BAR 0 Type	0 = I/O space 1 = Memory space	1	Integer
G15	C_IPIFBAR2PCIBAR_0	PCI BAR to which PLB BAR_0 is mapped	Valid PCIe address (2)	0xFFFF_FFFF	std_logic_vector
G16	C_IPIFBAR_1	PLB BAR_1 aperture low address	Valid PLB address (1)(3)(4) (5)	0xFFFF_FFFF	std_logic_vector
G17	C_IPIFBAR_HIGHADDR_1	PLB BAR_1 aperture high address	Valid PLB address (1)(3)(4)	0x0000_0000	std_logic_vector
G18	C_IPIFBAR_AS_1	PLB BAR_1 address size	0 = 32 bits 1 = 64 bits	0	Integer
G19	C_IPIFBAR_SPACE_TYPE_1	PLB BAR 1Type	0 = I/O space 1 = Memory space	1	Integer
G20	C_IPIFBAR2PCIBAR_1	PCI BAR to which PLB BAR_1 is mapped	Valid PCIe address (2)	0xFFFF_FFFF	std_logic_vector
G21	C_IPIFBAR_2	PLB BAR_2 aperture low address	Valid PLB address (1)(3)(4) (5)	0xFFFF_FFFF	std_logic_vector
G22	C_IPIFBAR_HIGHADDR_2	PLB BAR_2 aperture high address	Valid PLB address (1)(3)(4)	0x0000_0000	std_logic_vector
G23	C_IPIFBAR_AS_2	PLB BAR_2 address size	0 = 32 bits 1 = 64 bits	0	Integer
G24	C_IPIFBAR_SPACE_TYPE_2	PLB BAR 2 Type	0 = I/O space 1 = Memory space	1	Integer

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

Generic	Parameter Name	Description	Allowable Values	Default Value	VHDL Type
G25	C_IPIFBAR2PCIBAR_2	PCI BAR to which PLB BAR_2 is mapped	Valid PCIe address (2)	0xFFFF_FFFF	std_logic_vector
G26	C_IPIFBAR_3	PLB BAR_3 aperture low address	Valid PLB address (1)(3)(4)(5)	0xFFFF_FFFF	std_logic_vector
G27	C_IPIFBAR_HIGHADDR_3	PLB BAR_3 aperture high address	Valid PLB address (1)(3)(4)	0x0000_0000	std_logic_vector
G28	C_IPIFBAR_AS_3	PLB BAR_3 address size	0 = 32 bits 1 = 64 bits	0	Integer
G29	C_IPIFBAR_ SPACE_TYPE_3	PLB BAR 3 Type	0 = I/O space 1 = Memory space	1	Integer
G30	C_IPIFBAR2PCIBAR_3	PCI BAR to which PLB BAR_3 is mapped	Valid PCIe address (2)	0xFFFF_FFFF	std_logic_vector
G31	C_IPIFBAR_4	PLB BAR_4 aperture low address	Valid PLB address (1)(3)(4)(5)	0xFFFF_FFFF	std_logic_vector
G32	C_IPIFBAR_HIGHADDR_4	PLB BAR_4 aperture high address	Valid PLB address(1)(3)(4)	0x0000_0000	std_logic_vector
G33	C_IPIFBAR_AS_4	PLB BAR_4 address size	0 = 32 bits 1 = 64 bits	0	Integer
G34	C_IPIFBAR_SPACE_TYPE_4	PLB BAR 4 Type	0 = I/O space 1 = Memory space	1	Integer
G35	C_IPIFBAR2PCIBAR_4	PCI BAR to which PLB BAR_4 is mapped	Valid PCIe address (2)	0xFFFF_FFFF	std_logic_vector
G36	C_IPIFBAR_5	PLB BAR_5 aperture low address	Valid PLB address (1)(3)(4)(5)	0xFFFF_FFFF	std_logic_vector
G37	C_IPIFBAR_HIGHADDR_5	PLB BAR_5 aperture high address	Valid PLB address (1)(3)(4)	0x0000_0000	std_logic_vector
G38	C_IPIFBAR_AS_5	PLB BAR_5 address size	0 = 32 bits 1 = 64 bits	0	Integer
G39	C_IPIFBAR_SPACE_TYPE_5	PLB BAR 5 Type	0 = I/O space 1 = Memory space	1	Integer
G40	C_IPIFBAR2PCIBAR_5	PCI BAR to which PLB BAR_5 is mapped	Valid PCIe address (2)	0xFFFF_FFFF	std_logic_vector
G41	C_PCIBAR_NUM	Number of PCI address 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

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

Generic	Parameter Name	Description	Allowable Values	Default Value	VHDL Type
G42	C_PCIBAR_AS	Configures PCIBAR aperture width to be 32 bits wide or 64 bits wide	0 = Generates three 32-bit PCIBAR address apertures. 32-bit BAR example: PCIBAR_0 is 32 bits PCIBAR_1 is 32 bits PCIBAR_2 is 32 bits 1 = Generates three 64-bit PCIBAR address apertures. 64-bit BAR example: PCIBAR_0 and PCIBAR_1 concatenate to comprise 64-bit PCIBAR_0. PCIBAR_2 and PCIBAR_3 concatenate to comprise 64-bit PCIBAR_1. PCIBAR_4 and PCIBAR_5 concatenate to comprise 64-bit PCIBAR_2	1	Integer
G43	C_PCIBAR_LEN_0	Power of 2 in the size of bytes of PCI BAR_0 space	13-29	16	Integer
G44	C_PCIBAR2IPIFBAR_0	PLB BAR to which PCI BAR_0 is mapped	Valid PLB address	0x0000_0000	std_logic_vector
G45	C_PCIBAR_LEN_1	Power of 2 in the size of bytes of PCI BAR_1 space	13-29	16	Integer
G46	C_PCIBAR2IPIFBAR_1	PLB BAR to which PCI BAR_1 is mapped	Valid PLB address	0x0000_0000	std_logic_vector
G47	C_PCIBAR_LEN_2	Power of 2 in the size of bytes of PCI BAR_2 space	13-29	16	Integer
G48	C_PCIBAR2IPIFBAR_2	PLB BAR to which PCI BAR_2 is mapped	Valid PLB address	0x0000_0000	std_logic_vector
PCIe Core Configuration Parameters					
G49	C_NO_OF_LANES	Number of PCIe Lanes	1 - V5, V6, S6 4, 8 - V5 only	1	Integer
G50	C_DEVICE_ID	Device ID	16 bit vector	0x0000	std_logic_vector
G51	C_VENDOR_ID	Vendor ID	16 bit vector	0x0000	std_logic_vector
G52	C_CLASS_CODE	Class Code	24 bit vector	0x00_0000	std_logic_vector
G53	C_REV_ID	Rev ID	8 bit vector	0x00	std_logic_vector

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

Generic	Parameter Name	Description	Allowable Values	Default Value	VHDL Type
G54	C_SUBSYSTEM_ID	Subsystem ID	16 bit vector (valid only for Endpoint)	0x0000	std_logic_vector
G55	C_SUBSYSTEM_VENDOR_ID	Subsystem Vendor ID	16 bit vector (valid only for Endpoint)	0x0000	std_logic_vector
G56	C_PCIE_CAP_SLOT_ IMPLEMENTED	PCIE Capabilities Register Slot Implemented	1 = Downstream port is connected to add-in card slot 0 = Not add-in card slot (valid only for Root Complex)	1	Integer
G57	C_REF_CLK_FREQ	REFCLK input frequency	0 = 100 Mhz (required for V5) 1 = 125 Mhz (required for S6) 2 = 250 Mhz (required for V6)	0	Integer
IPIF Parameters					
G58	C_MPLB_DWIDTH	PLB Master Bus Data width	Automatically computed by platgen	Automatically computed by platgen	std_logic_vector
G59	C_MPLB_AWIDTH	PLB Master Bus Address width	32	32	std_logic_vector
G60	C_MPLB_SMALLEST_SLAVE	The data width of the smallest slave that will be accessing this IPIF	32/64/128	32	std_logic_vector
G61	C_MPLB_NATIVE_DWIDTH	Selects the Master IPIF data width	32 (for spartan6), 64 (for virtex5 or virtex6)	64	integer
G62	C_SPLB_DWIDTH	PLB Slave Bus Data width	Automatically computed by platgen	Automatically computed by platgen	std_logic_vector
G63	C_SPLB_AWIDTH	PLB Slave Bus Address width	32	32	std_logic_vector
G64	C_SPLB_MID_WIDTH	PLB Master ID Bus Width	3	log2(C_SPLB_ NUM_MASTER S)	Integer
G65	C_SPLB_NUM_MASTERS	Number of masters on the bus	1-8	1	Integer
G66	C_SPLB_SMALLEST_ MASTER	The data width of the smallest master that will be accessing this IPIF	32/64/128	32	std_logic_vector
G67	C_SPLB_NATIVE_DWIDTH	Selects the Slave IPIF data width	32 (for spartan6), 64 (for virtex5 or virtex6)	64	integer

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

Generic	Parameter Name	Description	Allowable Values	Default Value	VHDL Type
Non-HDL Generics					
G68	C_BOARD[6]	Automatically generates core level constraints for the selected board type. **This will not set the Pin LOC constraints**.	ml505, ml507, none	none	N/A
G69	C_DEVICE [6]	Used by the tools only. No user input required.	N/A	Set by Tools	N/A

1. This is a 32-bit address.
2. The width of this should match the address size (C_IPIFBAR_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.
4. The difference between C_IPIFBAR_n and C_IPIFBAR_HIGHADDR_n must be less than or equal to 0X1FFF_FFFF.
5. The minimum value of C_IPIFBAR must be greater than or equal to 0x1FFF.
6. This a NON_HDL generic used by the plbv46_pcie_v2_1_0.tcl to generate pcie core level constraints
7. The difference between C_ECAM_BASEADDR and C_ECAM_HIGHADDR must be less than or equal to 0x0FFF_FFFF and greater than or equal to 0x001F_FFFF.

Parameter Port Dependencies

Table 2 lists the effects of setting various generics.

Table 2: **Parameter Port Dependencies**

Generic	Parameter	Affects	Depends	Description
G1	C_FAMILY	G2, G3		
G2	C_SUBFAMILY		G1	Meaningful only if G1 = virtex5
G3	C_INCLUDE_RC	G6, G7, G14, G19, G24, G29, G34, G39	G1	Meaningful only if G1 = virtex6
G4	C_BASEADDR	G5	G5	G4 and G5 define range in PLB-memory space that is responded to by PLBv46 PCI Bridge register space
G5	C_HIGHADDR	G4	G4	G4 and G5 define range in PLB-memory space that is responded to by PLBv46 PCI Bridge register space
G6	C_ECAM_BASEADDR	G7	G3, G7	Meaningful only if G3 = 1 G6 and G7 define range in PLB-memory space that is responded to by the ECAM
G7	C_ECAM_HIGHADDR	G6	G3, G6	Meaningful only if G3 = 1 G6 and G7 define range in PLB-memory space that is responded to by the ECAM
G8	C_COMP_TIMEOUT			

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

Generic	Parameter	Affects	Depends	Description
G9	C_INCLUDE_BAROFFSET_REG	G15, G20, G25, G30, G40	G10	If G9 = 1 then G15, G20, G25, G30, G35 and G40 have no meaning. The number of registers included is set by G10
G10	C_IPIFBAR_NUM	G11 - G40		If G10 = 1, then G11 - G15 are enabled If G10 = 2, then G11 - G20 are enabled If G10 = 3, then G11 - G25 are enabled If G10 = 4, then G11 - G30 are enabled If G10 = 5, then G11 - G35 are enabled If G10 = 6, then G11 - G40 are enabled
G11	C_IPIFBAR_0	G12	G12	G11 and G12 define range in PLB-memory or I/O-space that is responded to by this device (IPIF BAR)
G12	C_IPIFBAR_HIGHADDR_0	G11	G11	G11 and G12 define range in PLB-memory or I/O-space that is responded to by this device (IPIF BAR)
G13	C_IPIFBAR_AS_0		G14	If G14 = 0, G13 must be = 0
G14	C_IPIFBAR_SPACE_TYPE_0	G13	G3	If G3 = 0, G14 must be = 1 If G3 = 1, G14 may be = 0 or 1
G15	C_IPIFBAR2PCIBAR_0		G9	Meaningful if G9 = 1
G16	C_IPIFBAR_1	G17	G17	G16 and G17 define range in PLB-memory or I/O-space that is responded to by this device (IPIF BAR)
G17	C_IPIFBAR_HIGHADDR_1	G16	G16	G16 and G17 define range in PLB-memory or I/O-space that is responded to by this device (IPIF BAR)
G18	C_IPIFBAR_AS_1		G19	If G19 = 0, G18 must be = 0
G19	C_IPIFBAR_SPACE_TYPE_1	G18	G3	If G3 = 0, G19 must be = 1 If G3 = 1, G19 may be = 0 or 1
G20	C_IPIFBAR2PCIBAR_1		G9	Meaningful if G9 = 1
G21	C_IPIFBAR_2	G22	G22	G21 and G22 define range in PLB-memory or I/O-space that is responded to by this device (IPIF BAR)
G22	C_IPIFBAR_HIGHADDR_2	G21	G21	G21 and G22 define range in PLB-memory or I/O-space that is responded to by this device (IPIF BAR)
G23	C_IPIFBAR_AS_2		G24	If G24 = 0, G23 must be = 0
G24	C_IPIFBAR_SPACE_TYPE_2	G23	G3	If G3 = 0, G24 must be = 1 If G3 = 1, G24 may be = 0 or 1
G25	C_IPIFBAR2PCIBAR_2		G9	Meaningful if G9 = 1
G26	C_IPIFBAR_3	G27	G27	G26 and G27 define range in PLB-memory or I/O-space that is responded to by this device (IPIF BAR)
G27	C_IPIFBAR_HIGHADDR_3	G26	G26	G26 and G27 define range in PLB-memory or I/O-space that is responded to by this device (IPIF BAR)

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

Generic	Parameter	Affects	Depends	Description
G28	C_IPIFBAR_AS_3		G29	If G29 = 0, G28 must be = 0
G29	C_IPIFBAR_SPACE_TYPE_3	G28	G3	If G3 = 0, G29 must be = 1 If G3 = 1, G29 may be = 0 or 1
G30	C_IPIFBAR2PCIBAR_3		G9	Meaningful if G9 = 1
G31	C_IPIFBAR_4	G32	G32	G31 and G32 define range in PLB-memory or I/O-space that is responded to by this device (IPIF BAR)
G32	C_IPIFBAR_HIGHADDR_4	G31	G31	G31 and G32 define range in PLB-memory or I/O-space that is responded to by this device (IPIF BAR)
G33	C_IPIFBAR_AS_4		G34	If G34 = 0, G33 must be = 0
G34	C_IPIFBAR_SPACE_TYPE_4	G33	G3	If G3 = 0, G34 must be = 1 If G3 = 1, G34 may be = 0 or 1
G35	C_IPIFBAR2PCIBAR_4		G9	Meaningful if G9 = 1
G36	C_IPIFBAR_5	G37	G37	G36 and G37 define range in PLB-memory or I/O-space that is responded to by this device (IPIF BAR)
G37	C_IPIFBAR_HIGHADDR_5	G36	G36	G36 and G37 define range in PLB-memory or I/O-space that is responded to by this device (IPIF BAR)
G38	C_IPIFBAR_AS_5		G39	If G39 = 0, G38 must be = 0
G39	C_IPIFBAR_SPACE_TYPE_5	G38	G3	If G3 = 0, G39 must be = 1 If G3 = 1, G39 may be = 0 or 1
G40	C_IPIFBAR2PCIBAR_5		G9	Meaningful if G9 = 1
G41	C_PCIBAR_NUM	G43 - G48	G3	If G3 = 0, G41 may be = 1, 2 or 3 If G3 = 1, G41 must be = 1 If G41 = 1, then G42, G43 are enabled If G41 = 2, then G42 - G46 are enabled If G41 = 3, then G42 - G48 are enabled
G42	C_PCIBAR_AS			
G43	C_PCIBAR_LEN_0			
G44	C_PCIBAR2IPIFBAR_0		G43	Only the high-order bits above the length defined by G43 are meaningful.
G45	C_PCIBAR_LEN_1			
G46	C_PCIBAR2IPIFBAR_1		G45	Only the high-order bits above the length defined by G45 are meaningful.
G47	C_PCIBAR_LEN_2			
G48	C_PCIBAR2IPIFBAR_2		G47	Only the high-order bits above the length defined by G47 are meaningful.

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

Generic	Parameter	Affects	Depends	Description
PCIe Core Configuration Parameters				

G49	C_NO_OF_LANES	TXP TXN
		RXP
		RXN
			G1	If G1 = spartan6 or virtex6 then
				G49 = 1 only
				if G1 = virtex5 then
				G49 = 1, 4 or 8.
				(Note: spartan6 is a fixed x1 lane Endpoint)

G50	C_DEVICE_ID			
G51	C_VENDOR_ID			
G52	C_CLASS_CODE			
G53	C_REV_ID			
G54	C_SUBSYSTEM_ID		G3	if G3 = 1 G54 is not meaningful
G55	C_SUBSYSTEM_VENDOR_ID		G3	if G3 = 1 G54 is not meaningful
G56	C_PCIE_CAP_SLOT_IMPLEMENTED		G3	if G3 = 0 G56 is not meaningful
G57	C_REF_CLK_FREQ		G1	if G1 = virtex5, G57 must be = 0
IPIF Parameters				
G58	C_MPLB_DWIDTH	G62	G62	G58 must be equal to G62
G59	C_MPLB_AWIDTH	G63	G63	G59 must be equal to G63
G60	C_MPLB_SMALLEST_SLAVE			
G61	C_MPLB_NATIVE_DWIDTH	G67	G1, G67	If G1 = spartan6 then G61 must be set to 32 if G1 = virtex5 or virtex6 then G61 must be 64 bits wide (Note: G61 and G67 must be equal width)
G62	C_SPLB_DWIDTH	G58	G58	G62 must be equal to G58
G63	C_SPLB_AWIDTH	G59	G59	G63 must be equal to G59
G64	C_SPLB_MID_WIDTH			
G65	C_SPLB_NUM_MASTERS			
G66	C_SPLB_SMALLEST_MASTER			
G67	C_SPLB_NATIVE_DWIDTH	G61	G1, G61	If G1 = spartan6 then G67 must be set to 32 if G1 = virtex5 or virtex6 then G67 must be 64 bits wide (Note: G61 and G67 must be equal width)
Non-HDL Generics				
G68	C_BOARD(6)			
G69	C_DEVICE(6)			

Top-level Interface Signals

The interface signals for the PLBv46 Bridge are described in Table 3. While there are independent clocks for the SPLB and MPLB interfaces these clocks are required to be synchronous and of the same frequency.

Table 3: **Top Level Interface Signals**

Signal Name	I/O	Description
Global Signals		
IP2INTC_Irpt	O	Interrupt signal
PLB Slave Interface		
SPLB_Clk	I	Slave PLB Clock [(1)]
SPLB_Rst	I	Slave PLB Reset
PLB_ABus [0:C_SPLB_AWIDTH-1]	I	PLB address bus
PLB_PAValid	I	PLB primary address valid indicator
PLB_masterID [0:C_SPLB_MID_WIDTH-1]	I	PLB current master identifier
PLB_abort	I	PLB abort bus request indicator
PLB_RNW	I	PLB read not write
PLB_BE [0:(C_SPLB_DWIDTH/8)-1]	I	PLB byte enables
PLB_MSize [0:1]	I	PLB master data bus size
PLB_size [0:3]	I	PLB transfer size
PLB_type [0:2]	I	PLB transfer type
PLB_wrDBus [0:C_SPLB_DWIDTH-1]	I	PLB write data bus
PLB_wrBurst	I	PLB burst write transfer indicator
PLB_rdBurst	I	PLB burst read transfer indicator
PLB_SAValid	I	PLB Secondary address valid
PLB_UABus[0:31]	I	PLB Upper address bus
PLB_BusLock	I	PLB Bus Lock
PLB_LockErr	I	PLB Lock Error
PLB_TAttribute[0:15]	I	PLB Attribute
PLB_RdPrim	I	PLB Read Primary
PLB_WrPrim	I	PLB Write Primary
PLB_RDPendPri[0:1]	I	PLB Read Pending on Primary
PLB_WrPendPri[0:1]	I	PLB Write Pending on Primary
PLB_RdPendReq	I	PLB Read Pending Request
PLB_WrPendReq	I	PLB Write Pending Request
Sl_addAck	O	Slave address acknowledge
Sl_SSize(0:1)	O	Slave data bus size
Sl_wait	O	Slave wait indicator
Sl_rearbitrate	O	Slave rearbitrate bus indicator

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

Signal Name	I/O	Description
Sl_wrDAck	O	Slave write data acknowledge
Sl_wrComp	O	Slave write transfer complete indicator
Sl_wrBTerm	O	Slave terminate write burst transfer
Sl_rdDBus(0:C_SPLB_DWIDTH-1)	O	Slave read data bus
Sl_rdWdAddr(0:3)	O	Slave read word address
Sl_rdDAck	O	Slave read data acknowledge
Sl_rdComp	O	Slave read transfer complete indicator
Sl_rdBTerm	O	Slave terminate read burst transfer
Sl_MBusy(0:C_SPLB_NUM_MASTERS-1)	O	Slave busy indicator
Sl_MRdErr(0:C_SPLB_NUM_MASTERS-1)	O	Slave read error indicator
Sl_MWrErr(0:C_SPLB_NUM_MASTERS-1)	O	Slave write error indicator
Sl_MIRQ(0:C_SPLB_NUM_MASTERS-1)	O	Slave Interrupt
PLB Master Interface		
MPLB_Clk	I	Master PLB Clock [(1)]
MPLB_Rst	I	Master PLB Reset
PLB_MAddrAck	I	PLB Master address acknowledge
PLB_MBusy	I	PLB Master slave busy indicator
PLB_MRdErr	I	PLB Master slave read error indicator
PLB_MWrErr	I	PLB Master slave write error indicator
PLB_MRdBTerm	I	PLB Master terminate read burst indicator
PLB_MRdDAck	I	PLB Master read data acknowledge
PLB_MRdDBus [0:C_MPLB_DWIDTH -1]	I	PLB Master read data bus
PLB_MRdWdAddr [0:3]	I	PLB Master read word address
PLB_MRearbitrate	I	PLB Master bus rearbitrate indicator
PLB_MSSize [0:1]	I	PLB Master slave data bus port width
PLB_MWrBTerm	I	PLB Master terminate write burst indicator
PLB_MWrDAck	I	PLB Master write data acknowledge
PLB_MTimeout	I	PLB Address Timeout
PLB_MIRQ	I	PLB interrupt
M_abort	O	Master abort bus request indicator
M_ABus[0:C_MPLB_AWIDTH-1]	O	Master address bus
M_BE[0:C_MPLB_DWIDTH/8-1]	O	Master byte enables
M_busLock	O	Master bus lock
M_MSize[0:1]	O	Master data bus port width
M_priority	O	Master bus request priority
M_rdBurst	O	Master burst read transfer indicator

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

Signal Name	I/O	Description
M_request	O	Master bus request
M_RNW	O	Master read not write
M_size[0:3]	O	Master transfer size
M_type	O	Master transfer type
M_wrBurst	O	Master burst write transfer indicator
M_wrDBus[0:C_MPLB_DWIDTH -1]	O	Master write data bus
M_LockErr	O	Master lock error
M_TAttribute[0:15]	O	Master Attribute
M_UABus[0:31]	O	Master Upper Address Bus
PCIe Interface		
REFCLK	I	PCIe Reference Clock
Bridge_Clk	O	125 MHz for Virtex-5 and Virtex-6 FPGAs 62.5 Mhz for Spartan-6 FPGAs (Used for debug only)
MSI_Request	I	Initiates a MSI write request
RXP[C_NO_OF_LANES-1 : 0]	I	RX serial interface
RXN[C_NO_OF_LANES-1 : 0]	I	RX serial interface
TXP[C_NO_OF_LANES-1 : 0]	O	TX serial interface
TXN[C_NO_OF_LANES-1 :0]	O	TX serial interface
Test Bench Debug Interface		
TB_Debug	O	bit 0 - indicates MRd TLP sent from bridge bit 1 - indicates MWr TLP sent from bridge bit 2 through 15 - reserved

1. `SPLB_Clk` and `MPLB_Clk` must be connected to the same clock source.

Memory Map

The memory map shown in Table 4 shows the address mapping for the PLBv46 Bridge. These registers are described in more detail in the following sections. All registers are offset from C_BASEADDR.

During a reset (either from the bus or from the IPIF register RESETMODULEREG) all registers are reset to their default values. The PCIe hard core registers are described in *LogiCORE IP Endpoint Block Plus v1.13 for PCI Express User Guide* (UG341), *LogiCORE IP Spartan-6 FPGA Integrated Endpoint Block v1.2 for PCI Express User Guide* (UG654) and *LogiCORE IP Virtex-6 FPGA Integrated Block v1.4 for PCI Express User Guide* (UG517).

Table 4: **Memory Map**

R/W	Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	Register Mnemonic
R/W	0x0000	Upper Address																															IPIFBAR2PCIBAR_0U
R/W	0x0004	Lower Address																															IPIFBAR2PCIBAR_0L
R/W	0x0008	Upper Address																															IPIFBAR2PCIBAR_1U
R/W	0x000C	Lower Address																															IPIFBAR2PCIBAR_1L
R/W	0x0010	Upper Address																															IPIFBAR2PCIBAR_2U
R/W	0x0014	Lower Address																															IPIFBAR2PCIBAR_2L
R/W	0x0018	Upper Address																															IPIFBAR2PCIBAR_3U
R/W	0x001C	Lower Address																															IPIFBAR2PCIBAR_3L
R/W	0x0020	Upper Address																															IPIFBAR2PCIBAR_4U
R/W	0x0024	Lower Address																															IPIFBAR2PCIBAR_4L
R/W	0x0028	Upper Address																															IPIFBAR2PCIBAR_5U
R/W	0x002C	Lower Address																															IPIFBAR2PCIBAR_5L
R/W	0x0030	Reserved																							BME	Reserved					E2	E1	E0	BCR
R/W-RC, RO-EP	0x0034	Reserved																Bus Number								Device No.				Function				PRIDR
RO	0x0038	Reserved																					Max Payload Size			Reserved					Max Read Request Size			PRCR
RO	0x003C	Reserved																						Link Width				LU	Reserved					PSR
R/TOW	0x0040	Reserved	SUR	MUR	MCA	MEP	SUC	MSI	SCT	SEP	SCA	SBO	NBE	LNKDN	Reserved				BME	Reserved														BIR
R/W	0x0044	Reserved	SUR	MUR	MCA	MEP	SUC	MSI	SCT	SEP	SCA	SBO	NBE	LNKDN	Reserved				BME	Reserved														BIER
R/W	0x0048	MSI Address																															MAR
RO	0x004C	MSI Data																															MDR
	0x0050 - 0x1FEC	Reserved																															Reserved
	PCIe Core Management Interface Registers																																	
R/W-RC, RO-EP	0x2000 - 0x3FFC	See UG341 LogiCORE IP Endpoint Block Plus for PCIe Users Guide, UG654 LogiCORE IP Spartan-6 FPGA Integrated Endpoint Block for PCI Express User Guide and UG517 LogiCORE IP Virtex-6 FPGA Integrated Block for PCI Express User Guide, in the "PCIe Configuration Space Header" table for a detailed description of these registers. To calculate the PLB address offset for a particular register, add 0x2000 to the register address.																															PCIeCore Registers
	Offset	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**Register Mnemonic**

PLB Base Address Translation Configuration Registers

The PLB Base Address Translation Configuration Registers and their offsets are shown in Table 5 and the register bits are described in Table 6. This set of registers can be used in two configurations based on the top level parameter C_IPIFBAR_AS_n. When the BAR is set to a 32-bit address space then the translation vector should be placed into the IPIFBAR2PCIBAR_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 IPIFBAR2PCIBAR_nU and the least significant 32 bits are written into IPIFBAR2PCIBAR_nL. When C_INCLUDE_BAR_OFFSET_REG = 1 these registers can be dynamically configured by software.

Table 5: **PLB Base Address Translation Configuration Registers**

Offset	Bits	Register Nmemonic
0x000	0-31	IPIFBAR2PCIBAR_0U
0x004	0-31	IPIFBAR2PCIBAR_0L
0x008	0-31	IPIFBAR2PCIBAR_1U
0x00C	0-31	IPIFBAR2PCIBAR_1L
0x010	0-31	IPIFBAR2PCIBAR_2U
0x014	0-31	IPIFBAR2PCIBAR_2L
0x018	0-31	IPIFBAR2PCIBAR_3U
0x01C	0-31	IPIFBAR2PCIBAR_3L
0x020	0-31	IPIFBAR2PCIBAR_4U
0x024	0-31	IPIFBAR2PCIBAR_4L
0x028	0-31	IPIFBAR2PCIBAR_5U
0x02C	0-31	IPIFBAR2PCIBAR_5L

Table 6: **PLB Base Address Translation Configuration Register Bit Definitions**

Bits	Name	Core Access	Reset Value	Description

0-31	Lower Address	R/W	if (C_IPIFBAR_AS_0 = 1) then
			reset value = C_IPIFBAR2PCIBAR_0(32 to 63)
			if (C_IPIFBAR_AS_0 = 0) then
			reset value = C_IPIFBAR2PCIBAR_0(0 to 31)
				Lower Address: To create the PCIe address, this is the value substituted for the least significant 32 bits of the PLB address.

0-31	Upper Address	R/W (if 64-bit address)	if (C_IPIFBAR_AS_0 = 1) then
			reset value = C_IPIFBAR2PCIBAR_0(0 to 31)
			if (C_IPIFBAR_AS_0 = 0) then
			reset value = 0x00000000
				Upper Address: To create the PCIe address, this is the value substituted for the most significant 32 bits of the PLB address.

0-31	Lower Address	R/W	if (C_IPIFBAR_AS_1 = 1) then
			reset value = C_IPIFBAR2PCIBAR_1(32 to 63)
			if (C_IPIFBAR_AS_1 = 0) then
			reset value = C_IPIFBAR2PCIBAR_1(0 to 31)
				Lower Address: To create the PCIe address, this is the value substituted for the least significant 32 bits of the PLB address.

Table 6: **PLB Base Address Translation Configuration Register Bit Definitions *(Cont'd)***

Bits	Name	Core Access	Reset Value	Description

0-31	Upper Address	R/W
		(if 64-bit address)
			if (C_IPIFBAR_AS_1 = 1) then
			reset value = C_IPIFBAR2PCIBAR_1(0 to 31)
			if (C_IPIFBAR_AS_1 = 0) then
			reset value = 0x00000000
				Upper Address: To create the PCIe address, this is the value substituted for the most significant 32 bits of the PLB address.

0-31	Lower Address	R/W	if(C_IPIFBAR_AS_2 = 1) then
			reset value = C_IPIFBAR2PCIBAR_2(32 to 63)
			if (C_IPIFBAR_AS_2 = 0) then
			reset value = C_IPIFBAR2PCIBAR_2(0 to 31)
				Lower Address: To create the PCIe address, this is the value substituted for the least significant 32 bits of the PLB address.

0-31	Upper Address	R/W
		(if 64-bit address)
			if (C_IPIFBAR_AS_2 = 1) then
			reset value = C_IPIFBAR2PCIBAR_2(0 to 31)
			if (C_IPIFBAR_AS_2 = 0) then
			reset value = 0x00000000
				Upper Address: To create the PCIe address, this is the value substituted for the most significant 32 bits of the PLB address.

0-31	Lower Address	R/W	if (C_IPIFBAR_AS_3 = 1) then
			reset value = C_IPIFBAR2PCIBAR_3(32 to 63)
			if (C_IPIFBAR_AS_3 = 0) then
			reset value = C_IPIFBAR2PCIBAR_3(0 to 31)
				Lower Address: To create the PCIe address, this is the value substituted for the least significant 32 bits of the PLB address (bits 32 to 63).

0-31	Upper Address	R/W
		(if 64-bit address)
			if (C_IPIFBAR_AS_3 = 1) then
			reset value = C_IPIFBAR2PCIBAR_3(0 to 31)
			if (C_IPIFBAR_AS_3 = 0) then
			reset value = 0x00000000
				Upper Address: To create the PCIe address, this is the value substituted for the most significant 32 bits of the PLB address.

0-31	Lower Address	R/W	if (C_IPIFBAR_AS_4 = 1) then
			reset value = C_IPIFBAR2PCIBAR_4(32 to 63)
			if (C_IPIFBAR_AS_4 = 0) then
			reset value = C_IPIFBAR2PCIBAR_4(0 to 31)
				Lower Address: To create the PCIe address, this is the value substituted for the least significant 32 bits of the PLB address.

0-31	Upper Address	R/W
		(if 64-bit address)
			if (C_IPIFBAR_AS_4 = 1) then
			reset value = C_IPIFBAR2PCIBAR_4(0 to 31)
			if (C_IPIFBAR_AS_4 = 0) then
			reset value = 0x00000000
				Upper Address: To create the PCIe address, this is the value substituted for the most significant 32 bits of the PLB address.

0-31	Lower Address	R/W	if (C_IPIFBAR_AS_5 = 1) then
			reset value = C_IPIFBAR2PCIBAR_5(32 to 63)
			if (C_IPIFBAR_AS_5 = 0) then
			reset value = C_IPIFBAR2PCIBAR_5(0 to 31)
				Lower Address: To create the PCIe address, this is the value substituted for the least significant 32 bits of the PLB address.

0-31	Upper Address	R/W
		(if 64-bit address)
			if (C_IPIFBAR_AS_5 = 1) then
			reset value = C_IPIFBAR2PCIBAR_5(0 to 31)
			if (C_IPIFBAR_AS_5 = 0) then
			reset value = 0x00000000
				Upper Address: To create the PCIe address, this is the value substituted for the most significant 32 bits of the PLB address.

Bridge Control Register (BCR, Offset 0x30)

The Bridge Control Register shown in Figure 2 enables the operation of the slave bridge and master bridge via the Bridge Control Register bits BME, E0-E2.

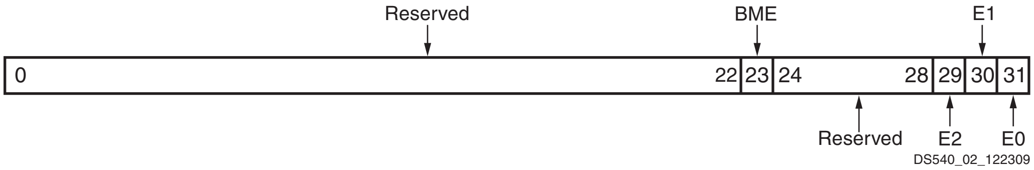

Figure 2: **Bridge Control Register**

Table 7: **Bridge Control Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0-22	Reserved			
23	BME	R/W	0b	**Bus Master Enable**: This is used to enable the PCIe bus master capability and completion capability. This will prevent both the requests from being sourced by the PCIe hard core and the PLB slave read completions from being returned to a PCIe requester if set to 0. PLB master requests will be re-arbitrated on the PLB when set to 0. Please note that the Bus Maser Enable bit of the PCI Configuration space will also stop the PLB slave read completions from being returned to a PCIe requester if set to 0.
24-28	Reserved			
29-31	E0, E1, E2	R/W	000000b	**Enable PCIe BAR 0-2**: This is used to enable the PCIe BARs after SW configuration. This allows proper TLP address filtering and translation to occur. Active high. Bit 31- BAR0 enable Bit 30- BAR1 enable Bit 29- BAR2 enable

PCIe Requester ID Register (PRIDR, Offset 0x34)

The PCIe Requester ID Register shown in Figure 3 provides the Bus Number, Device Number and Function Number of the Bridge. When the bridge is configured as a Root Complex (C_INCLUDE_RC = 1), the Bus Number and Device Number values are writable. This register provides the Requester ID for all TLPs generated by the slave bridge, including the Power Limit Message, which is automatically sent, after link up, when the BME bit is set in the Bridge Control Register (BCR). Therefore, the self-configuration SW must set the PRIDR register before setting the BME bit, or the Power Limit Message may be sent with the wrong requester ID. When the bridge is configured as an Endpoint (C_INCLUDE_RC = 0), the register is read only and is only valid when LinkUp = 1.

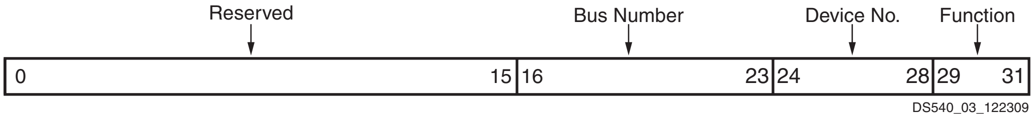

Figure 3: **PCIe Requester ID Register**

Table 8: **PCIe Requester ID Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0-15	Not Used			
16-23	Bus Number	RO for Endpoint, R/W for Root Complex	00h	**Bus Number**: When configured as an Endpoint (C_INCLUDE_RC = 0), this is the bus number of the Bridge assigned by the Root Complex upon enumeration. When configured as a Root Complex (C_INCLUDE_RC = 1), this is where the bus number of the Bridge must be written by the user.
24-28	Device Number	RO for Endpoint, R/W for Root Complex	0h	**Device Number**: When configured as an Endpoint (C_INCLUDE_RC = 0), this is the device number of the Bridge assigned by the Root Complex upon enumeration. When configured as a Root Complex (C_INCLUDE_RC = 1), this is where the device number of the Bridge must be written by the user.
29-31	Function Number	RO	000b	**Function Number**: This is the function number of the Bridge. This value is hard coded to “000” inside the Bridge.

PCIe Request Control Register (PRCR, Offset 0x38)

The PCIe Requester Control Register shown in Figure 4 shows the max payload size and max read request size status from the hard block. PLBv46 Bridge limits the max payload size to 1024 bytes for Virtex-6 FPGAs and 512 bytes for Virtex-5 and Spartan-6 FPGAs, when performing PLB to PCIe memory transactions. The maximum read request size is 4096 bytes when performing PCIe to PLB memory read requests. The data in this register is only valid when LinkUp = 1.

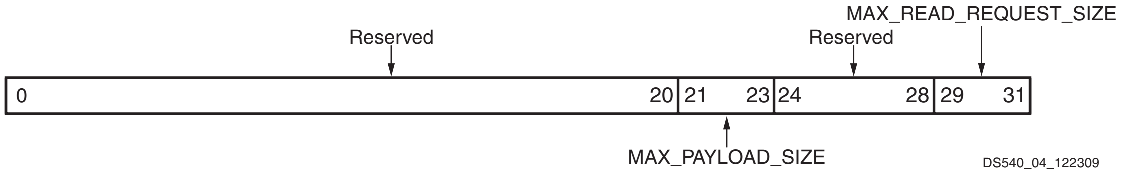

Figure 4: **PCIe Request Control Register**

Table 9: **PCIe Request Control Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0-20	Reserved			
21-23	Max Payload Size	RO	000b	**MAX PAYLOAD SIZE**: This value is set by the RC. 000 = 128 001 = 256 010 = 512 011 = 1024 (Virtex-6 only) 100 = Reserved 101 = Reserved 110 = Reserved 111 = Reserved

Table 9: **PCIe Request Control Register Bit Definitions *(Cont'd)***

Bit(s)	Name	Core Access	Reset Value	Description
24-28	Reserved			
29-31	Max Read Request Size	RO	010b	**MAX READ REQUEST SIZE**: This value is set by the RC. 000 = 128 001 = 256 010 = 512 011 = 1024 100 = 2048 101 = 4096 110 = Reserved 111 = Reserved

PCIe Status Register (PSR, Offset 0x3C)

The PCIe Status Register shown in Figure 5 holds the status outputs from the PCIe core. Table 10 provides the bit description of the PCIe core status outputs. This register is only accessible when LinkUp = 1.

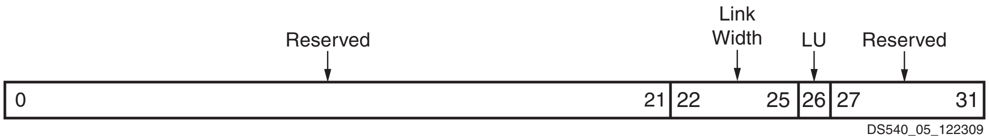

Figure 5: **PCIe Status Register**

Table 10: **PCIe Status Register Bit Definitions**

Bits	Name	Core Access	Reset Value	Description
0-21	Reserved			
22-25	Link Width	RO	C_NO_OF_LANES	**Negotiated Link Width (22 to 25)**: 0001 = One Lane 0100 = Four Lane (V5 only) 1000 = Eight Lane (V5 only)
26	LU	RO	0b	**Link Up:** This bit is set when link training is complete and the link is operational.
27-31	Reserved			

Bridge Interrupt Register (BIR, Offset 0x40)

The Bridge Interrupt Register shown in Figure 6 is used to send Bridge interrupts to the host processor.

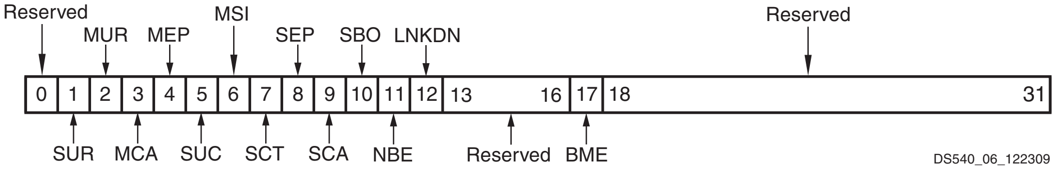

Figure 6: **Bridge Interrupt Register**

Table 11: **Bridge Interrupt Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0	Reserved			
1	SUR	R/TOW	0b	**Slave Unsupported Request**: Asserted when the slave side of the Bridge detects a completion TLP with completion status = supported request from PCIe.
2	MUR	R/TOW	0b	**Master Unsupported Request**: Asserted when the master side of the Bridge detects a completion TLP with completion status = unsupported request from PCIe.
3	MCA	R/TOW	0b	**Master Completion Abort**: Asserted when the master side of the Bridge has received a timeout from the PLB bus.
4	MEP	R/TOW	0b	**Master Error Poison**: Asserted when the master side of the Bridge has detected a TLP with the poison bit set from PCIe.
5	SUC	R/TOW	0b	**Slave Unexpected Completion**: Asserted when the slave side of the Bridge detects an unexpected completion TLP from PCIe.
6	MSI	R/TOW	0	**Message Signaled Interrupt:** Asserted when the master side of the bridge has received an MSI MemWr TLP. Implemented only when C_INCLUDE_RC=1.
7	SCT	R/TOW	0b	**Slave Completion Timeout:** Asserted when the slave side of the Bridge has detected a completion timeout
8	SEP	R/TOW	0b	**Slave Error Poison**: Asserted when the slave side of the Bridge has detected a TLP with the poison bit set from PCIe.
9	SCA	R/TOW	0b	**Slave Completion Abort**: Asserted when the slave side of the Bridge has detected a completion TLP with the status = completion abort from PCIe.
10	SBO	R/TOW	0b	**Slave BAR Overrun**: Asserted when the slave side of the Bridge has detected PLB request with an address outside the BAR address range.
11	NBE	R/TOW	0b	**Non-Contiguous Byte Enables**: Asserted when the slave side of the Bridge detects a write request from PCIe with non contiguous byte enables.
12	LNKDN	R/TOW	0b	**Link Down**: Asserted when the PCI Express link goes down. Note: The LNKDN bit will be asserted after power up and will need to be cleared by software.
13-16	Reserved			
17	BME	R/TOW	0b	**Bus Master Enable**: Asserted when the BME bit is set in the Command register of the PCI configuration header to enable the Bridge to send memory requests. As an EP, this typically happens when an external Root Complex does enumeration. As an RC, this happens during self-configuration.
18-31	Reserved			

Bridge Interrupt Enable Register (BIER, Offset 0x44)

The Bridge Interrupt Enable Register shown in Figure 7 is used to enable/disable (or mask) the Bridge status bits to the Bridge Interrupt Register (offset 0x40).

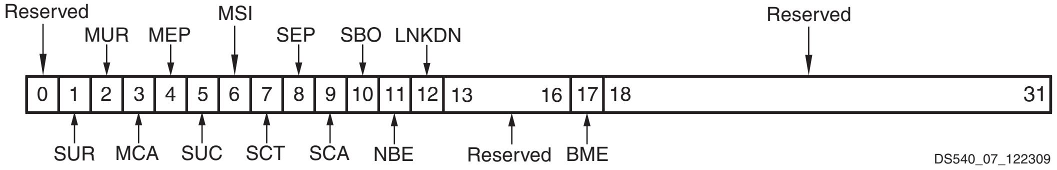

Figure 7: **Bridge Interrupt Enable Register**

Table 12: Bridge Interrupt Enable Register

Bit(s)	Name	Core Access	Reset Value	Description
0	Reserved			
1	SUR	R/W	0b	**Slave Unsupported Request Enable**: 1- Enables Interrupt 0- Disables Interrupt
2	MUR	R/W	0b	**Master Unsupported Request Enable**: 1- Enables Interrupt 0- Disables Interrupt
3	MCA	R/W	0b	**Master Completion Abort Enable**: 1- Enables Interrupt 0- Disables Interrupt
4	MEP	R/W	0b	**Master Error Poison Enable**: 1- Enables Interrupt 0- Disables Interrupt
5	SUC	R/W	0b	**Slave Unexpected Completion Enable**: 1- Enables Interrupt 0- Disables Interrupt
6	MSI	R/W	0	**Message Signaled Interrupt Enable:** 1- Enables Interrupt 0- Disables Interrupt Implemented only when C_INCLUDE_RC=1
7	SCT	R/W	0b	**Slave Completion Timeout Enable**: 1- Enables Interrupt 0- Disables Interrupt
8	SEP	R/W	0b	**Slave Error Poison Enable**: 1- Enables Interrupt 0- Disables Interrupt
9	SCA	R/W	0b	**Slave Completion Abort Enable**: 1- Enables Interrupt 0- Disables Interrupt
10	SBO	R/W	0b	**Slave BAR Overrun Enable**: 1- Enables Interrupt 0- Disables Interrupt

Table 12: **Bridge Interrupt Enable Register**

Bit(s)	Name	Core Access	Reset Value	Description
11	NBE	R/W	0b	**Non-Contiguous Byte Enables Enable**: 1- Enables Interrupt 0- Disables Interrupt
12	LNKDN	R/W	0b	**Link Down Enable**: 1- Enables Interrupt 0- Disables Interrupt
13-16	Reserved			
17	BME	R/W	0b	**Bus Master Enable**: 1- Enables Interrupt 0- Disables Interrupt
18-31	Reserved			

MSI Address Register (MAR, Offset 0x48)

The MSI Address Register shown in Table 13 provides the address value used to detect an MSI MemWr TLP when it is received from the PCIe Bus. It is required that a value outside the bar range is written to this register to uniquely decode a MSI TLP. The register value is compared to the address contained in received MemWr TLPs. When a matching value is detected, the MSI interrupt is asserted and the data is written to the MDR. Only the 32-bit address version of MSI is supported. Note that if for some reason a value within the bar range is written to this register, it will cause the MemWr TLP to be treated as both a typical MemWr TLP and a MSI interrupt requesting MemWr TLP, resulting in spurious MSI interrupt generation.

Table 13: **MSI Address Register Bit definitions**

Bits	Name	Core Access	Reset Value	Description
31-0	MSI Address	R/W	0000_0000h	**MSI Address**: The 32-bit PCIe address used to compare with received PCIe address in an MSI MemWr TLP.

MSI Data Register (MDR, Offset 0x4C)

The MSI Data Register shown in Table 14 provides the data value that is contained in an MSI MemWr TLP when it is received from the PCIe Bus. When the conditions are met for generating an MSI interrupt, as defined in the preceding MSI Address Register description, the single dword (32-bit) payload contained in the MSI MemWr TLP is written into this register.

Table 14: **MSI Data Register Bit definitions**

Bits	Name	Core Access	Reset Value	Description
31-0	MSI Data	R	0000_0000h	**MSI Data**: The data from the last received MSI MemWr TLP.

Clock and Reset Interface

Clock Interface

Xilinx recommends using the 100 MHz differential clock from the host PCI Express connector edge and connecting the differential inputs to a differential to single ended utility core. The output of the utility core needs to connect to the REFCLK input of the Bridge. The C_REF_CLK_FREQ parameter must be set to 100 Mhz if this clock is used.

Reset Interface

The `MPLB_Rst` and `SPLB_Rst` must be asserted simultaneously and held for a minimum of 300 ns. When using the EDK tools to build a system, the user is encouraged to connect the PERSTN pin of the host PCIe connector to the `aux_reset_in` port of the Proc_Sys_Reset module. The `bus_struct_reset` output of the Proc_Sys_Reset module must then be connected to the sys_rst input of the PLBv46 bus module. The assignment of the PLBv46 bus module instance label to the plbv46_pcie BUS_INTERFACE_SPLB and BUS_INTERFACE_MPLB will cause the PLBv46 bus module `plb_rst` output to be connected to the `MPLB_Rst` and `SPLB_Rst` of the Bridge for proper reset operation

Link Down and Hot Reset as Endpoint

When the Link goes down or when a Hot Reset is received the hard core will be reset and the internal LinkUp output from the hard core will deassert. When the PLBv46 Bridge detects the deassertion of LinkUp from the hard core the Bridge will discard any ingress/egress TLPs that were in transit and will set the LinkDown interrupt in the Bridge Interrupt Register. The PCIe hard core, internal bridge registers and bridge state machines are not reset by the Hot Reset. All PLB requests to the PLBv46 Bridge are re-arbitrated when the link is down

Generating Hot Reset as Root Complex

Hot Reset can be generated by a write of 1 to the Bridge Control register bit 6 within the Type 1 Configuration Space Header via the management interface. Initiating the Hot Reset causes the link to go down and link train. This bit is automatically cleared once the link is down. When the PLBv46 Bridge detects the deassertion of LinkUp from the hard core the Bridge will discard any ingress/egress TLPs that were in transit and will set the LinkDown interrupt in the Bridge Interrupt Register. The PCIe hard core, internal bridge registers and bridge state machines are not reset by the Hot Reset. Hot Resets do not go upstream, so a switch cannot send a Hot Reset to the bridge when it is configured as a Root Complex. After a Hot Reset, initialization must start within 80 ms. Self configuration is not needed since the bridge and PCIe hard core are not reset. All PLB requests to the PLBv46 Bridge are re-arbitrated when the link is down.

Enhanced Configuration Access Mechanism

The PLBv46 Bridge implements the PCI Express Enhanced Configuration Access Mechanism (ECAM) when configured with Root Complex capability. This mechanism uses a memory-mapped address space, as defined by the C_ECAM_BASEADDR and C_ECAM_HIGHADDR parameters. Read and write accesses to this address space are mapped to PCI Express Configuration space as defined in Table 15. The PLB address is used to define the various fields in the Configuration read and write request Packet headers. The PCIe Bus Number(s) supported by the bridge as a Root Complex is defined by the size of the number of address bits n allocated to the Bus Number field in the ECAM address space. For example: with $n = 2$, there are $2^2 = 4$ possible PCIe Bus Numbers. The value of n is implied by the ECAM range as defined by the C_ECAM_BASEADDR and C_ECAM_HIGHADDR parameters.

The BME bit of the Bridge Control Register (BCR) must be enabled for ECAM access. ECAM memory space is to be accessed only by single data beat transactions. If burst transactions are attempted, the behavior is not defined. It is the responsibility of software to insure that a configuration write has completed before attempting a configuration read.

When an ECAM access is attempted to a bus number that is out of the bus_number and subordinate bus number set by the platform specific code within the Type 1 Configuration Space Header, the bridge will not stop the generated configuration request. In the case of a write request, the bridge will set an SUR interrupt if an UR status is received in the completion. If no completion was received, the bridge will set an STO interrupt. For the read request case, see Configuration Read Response in the Slave Side Abnormal Conditions section.

Table 15: **Enhanced Configuration Address Mapping**

PLB Address	PCI Express Configuration Space
[12 – n :11]	Bus Number $1 \le n \le 8$
[12:16]	Device Number
[17:19]	Function Number
[20:23]	Extended Register Number
[24:29]	Register Number
[30:31]	Along with size of the access, used to generate Byte Enables

Platform Specific SW Requirements

As part of the root complex functionality and before the enumeration procedure for remote devices start, platform specific SW must set type 1 configuration space registers, the Requester ID register, and the BME bit of the bridge control register (BCR). This is "self-configuration" and is performed via the memory mapped range offset 0x2000-0x3fff and the offset 0x34 PRIDR register. The BME bit must be set after the Requester ID register is set in order for the Power Limit Message to contain the set Requester ID register value. As with the EP configuration, PCIe to PLB transactions after enumeration must be enabled by bits in the BCR as well. The required order is as follows:

1. Self-configure via the management interface
2. Set the Requester ID register value
3. Set BCR
4. Enumerate via ECAM

Address Translation

PCIe address space is different than PLB address space and to access one address space from another address space requires an address translation process.

The PLB side the bridge supports up to six 32-bit or 64-bit IPIF base address registers (BARs) and the generics used to configure the BARs are C_IPIFBAR_NUM, C_IPIFBAR_n, C_IPIFBAR_HIGHADDR_n, C_IPIFBAR2PCIBAR_n and C_IPIFBAR_AS_n, where "n" represents a particular IPIF BAR number from 0 to 5. The PCIe side the bridge supports up to three 64-bit PCI BARs and the generics used to configure the BARs are C_PCIBAR_NUM, C_PCI2IPIFBAR_n and C_PCIBAR_LEN_n, where "n" represents a particular PCIe BAR number from 0 to 2. The C_INCLUDE_BAROFFSET_REG generic allows for dynamic address translation. When this parameter is set to one, the IPIFBAR2PCIBAR_n translation vectors can be changed via software.

In the 4 examples to follow, Example 1 demonstrates how to set up four 32-bit IPIF BARs and translate the PLB address to a PCIe address. Example 2 demonstrates how to set up three 64-bit IPIF BARs and translate the PLB address to a PCIe address. Example 3 demonstrates how to set up two 64-bit PCIe BARs and translate the PCI address to a PLB address. Example 4 demonstrates how set up a combination of two 32-bit IPIF BARs and two 64-bit IPIF BARs and translate the PLB address to PCIe address.

Example 1

This example shows the generic settings to set up 4 independent 32-bit IPIF BARs and address translation of PLB addresses to a remote PCIe address space. Note that this setting of IPIF BARs does not depend on the PCIe BARs within the Bridge.

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

```
C_IPIFBAR_AS_0=0
C_IPIFBAR_0=0x12340000
C_IPIF_HIGHADDR_0=0x1234FFFF
C_IPIFBAR2PCIBAR_0=0x5671XXXX (Bits 16-31 are don't cares)

C_IPIFBAR_AS_1=0
C_IPIFBAR_1=0xABCDE000
C_IPIF_HIGHADDR_1=0xABCDFFFF
C_IPIFBAR2PCIBAR_1=0xFEDC0XXX (Bits 19-31 are don't cares)

C_IPIFBAR_AS_2=0
C_IPIFBAR_2=0xFE000000
C_IPIF_HIGHADDR_2=0xFFFFFFFF
C_IPIFBAR2PCIBAR_2=0x40XXXXXX (Bits 7-31 are don't cares)

C_IPIFBAR_AS_3=0
C_IPIFBAR_3=0x00000000
C_IPIF_HIGHADDR_3=0x0000007F
C_IPIFBAR2PCIBAR_3=0x8765438X (Bits 25-31 are don't cares)
```

Accessing the Bridge IPIFBAR_0 with address `0x12340ABC` on the PLB bus yields `0x56710ABC` on the PCIe bus.

Accessing the Bridge IPIFBAR_1 with address `0xABCDF123` on the PLB bus yields `0xFEDC1123` on the PCIe bus.

Accessing the Bridge IPIFBAR_2 with address `0xFFFEDCBA` on the PLB bus yields `0x41FEDCBA` on the PCIe bus.

Accessing the PLBv46 Bridge IPIFBAR_3 with address `0x00000071` on the PLB bus yields `0x876543F1` on the PCIe bus.

Example 2

This example shows the generic settings to set up 3 independent 64-bit IPIF BARs and address translation of PLB addresses to a remote PCIe address space. Note that this setting of IPIF BARs does not depend on the PCIe BARs within the Bridge.

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

```
C_IPIFBAR_AS_0=1
C_IPIFBAR_0=0x12340000
C_IPIF_HIGHADDR_0=0x1234FFFF
C_IPIFBAR2PCIBAR_0=0x500000005671XXXX (Bits 48-63 are don't cares)

C_IPIFBAR_AS_1=1
C_IPIFBAR_1=0xABCDE000
C_IPIF_HIGHADDR_1=0xABCDFFFF
```

```
C_IPIFBAR2PCIBAR_1=0x60000000FEDC0XXX (Bits 51-63 are don't cares)

C_IPIFBAR_AS_2=1
C_IPIFBAR_2=0xFE000000
C_IPIF_HIGHADDR_2=0xFFFFFFFF
C_IPIFBAR2PCIBAR_2=0x7000000040XXXXXX (Bits 39-63 are don't cares)
```

Accessing the Bridge IPIFBAR_0 with address `0x12340ABC` on the PLB bus yields `0x5000000056710ABC` on the PCIe bus.

Accessing the Bridge IPIFBAR_1 with address `0xABCDF123` on the PLB bus yields `0x60000000FEDC1123` on the PCIe bus.

Accessing the Bridge IPIFBAR_2 with address `0xFFFEDCBA` on the PLB bus yields `0x7000000041FEDCBA` on the PCIe bus.

Example 3

This example shows the generic settings to set up 2 independent PCIe BARs and address translation of PCIe addresses to a remote PLB address space. Note that this setting of PCIe BARs does not depend on the IPIF BARs within the Bridge. Note that the C_PCIBAR_LEN_n parameter has a maximum value of 29, thus the maximum allowable address space is 2^{29} = 512M per BAR.

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

```
BAR 0 is set to 0x20000000ABCDE800 by the Root Complex
C_PCIBAR_LEN_0=11
C_PCIBAR2IPIFBAR_0=0x123450XX (Bits 21-31 are don't cares)

BAR 1 is set to 0xA000000012000000 by Root Complex
C_PCIBAR_LEN_1=25
C_PCIBAR2IPIFBAR_1=0xFEXXXXXX (Bits 7-31 are don't cares)
```

Accessing the Bridge PCIBAR_0 with address `0x20000000ABCDEFF4` on the PCIe bus yields `0x123457F4` on the PLB bus.

Accessing Bridge PCIBAR_1 with address `0xA00000001235FEDC` on the PCIe bus yields `0xFE35FEDC` on the PLB bus.

Example 4

This example shows the generic settings to set up a combination of 2 independent 32-bit IPIF BARs and 2 independent 64-bit BARs and address translation of PLB addresses to a remote PCIe address space. Note that this setting of IPIF BARs does not depend on the PCIe BARs within the Bridge.

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

```
C_IPIFBAR_AS_0=0
C_IPIFBAR_0=0x12340000
C_IPIF_HIGHADDR_0=0x1234FFFF
C_IPIFBAR2PCIBAR_0=0x5671XXXX (Bits 16-31 are don't cares)

C_IPIFBAR_AS_1=1
C_IPIFBAR_1=0xABCDE000
C_IPIF_HIGHADDR_1=0xABCDFFFF
C_IPIFBAR2PCIBAR_1=0x50000000FEDC0XXX (Bits 51-63 are don't cares)

C_IPIFBAR_AS_2=0
C_IPIFBAR_2=0xFE000000
```

```
C_IPIF_HIGHADDR_2=0xFFFFFFFF
C_IPIFBAR2PCIBAR_2=0x40XXXXXX (Bits 7-31 are don't cares)

C_IPIFBAR_AS_3=1
C_IPIFBAR_3=0x00000000
C_IPIF_HIGHADDR_3=0x0000007F
C_IPIFBAR2PCIBAR_3=0x600000008765438X (Bits 57-63 are don't cares)
```

Accessing the Bridge IPIFBAR_0 with address `0x12340ABC` on the PLB bus yields `0x56710ABC` on the PCIe bus.

Accessing the Bridge IPIFBAR_1 with address `0xABCDF123` on the PLB bus yields `0x50000000FEDC1123` on the PCIe bus.

Accessing the Bridge IPIFBAR_2 with address `0xFFFEDCBA` on the PLB bus yields `0x41FEDCBA` on the PCIe bus.

Accessing the PLBv46 Bridge IPIFBAR_3 with address `0x00000071` on the PLB bus yields `0x60000000876543F1` on the PCIe bus.

Interrupts

The Bridge has two interrupt mechanisms to prompt devices external to the Bridge of certain events. The first is the `IP2INTC_Irpt` interrupt pin and the second is the MSI_REQUEST pin.

The `IP2INTC_Irpt` pin can be configured to send interrupts based on the settings of the Bridge Interrupt Enable register. The `IP2INTC_Irpt` signals interrupts to devices attached to the PLB side of the Bridge. Note that one of the interrupts defined in the Bridge Interrupt Enable register is to indicate of the receipt of a Message Signaled Interrupt when the bridge is operating in Root Complex mode (C_INCLUDE_RC=1).

The MSI_REQUEST pin is used to trigger a Message Signaled Interrupt via a special MemWr TLP to an external PCIe Root Complex on the PCIe side of the Bridge. The MSI_REQUEST pin input is level detected and the pin must see a level high (level = 1) for a minimum of two PLB_Clks before the Bridge sends the Message Signaled Interrupt. The address and data contained in this MemWr TLP are determined by configuration of registers within the PCIe hardblock by an external PCIe Root Complex. Note that this MSI_REQUEST pin input is valid only when the bridge is operating in Endpoint mode (C_INCLUDE_RC=0).

Unexpected Completion

When the slave bridge receives a completion TLP it matches the header Address, Length and Tag to the outstanding requested Address and Tag. A match failure indicates the TLP is an Unexpected Completion which will result in the completion TLP being discarded and a Slave Unexpected Completion (SUC) interrupt being set. Normal operation then continues.

Malformed TLP

If a malformed TLP is received, the PCIe hard core identifies the request, then removes it. The Bridge will not see the TLP.

Abnormal Conditions

This section describes how the Slave side and Master side of the PLBv46 Bridge handle abnormal conditions.

Slave Side Abnormal Conditions

Slave side abnormal conditions are classified into two groups: 1), Bar Length Overrun errors and 2), Completion TLP Errors. The following sections describe the manner in which the Bridge will handle the error groups.

BAR Length Overrun

The Slave side of the Bridge monitors PLB read and write requests to ensure that the request is within the Base Address Register (BAR) address range. Any PLB to PCIe read or write request that starts within a valid BAR range but ends outside the valid BAR range, will cause the Bridge to issue a Slave BAR Overrun (SBO) interrupt to the Bridge Interrupt Status Register. The Bridge will also assert `PLB_MRdErr` for a read request on the PLB or `PLB_MWrErr` for a write request on the PLB.

Completion TLP Errors

Any request to the PCIe bus (except for posted Memory write) requires a completion TLP to complete the associated PLB request. The Slave side of the PLBv46 Bridge checks the received completion TLPs for errors and checks for completion TLPs that are never returned (Timeout). Each of the completion TLP error types are discussed in the subsequent sections. When the Slave side of the PLBv46 Bridge detects completion TLP error (or Timeout), it discards the erred completion TLP, issues an interrupt to the Bridge Interrupt Status register, automatically reissues the request to the PCIe bus, then rearbitrates the PLB. When the Slave side of the PLBv46 Bridge detects two back-to-back completion TLP errors (or Timeouts), it discards the first erred completion TLP, issues an interrupt to the Bridge Interrupt Status register, reissues the PCIe TLP request, then rearbitrates the PLB. When the Slave side receives the second erred completion TLP (or Timeout), it issues an interrupt to the Bridge Interrupt Status register and, in the case of a Memory read or IO read, asserts PLB_MRdErr for the read request on the PLB.

Unsupported Request

A PCIe device may not be capable of satisfying a specific read request. For example, the read request targets an unsupported PCIe address causing the PCIe completer to return a completion TLP with a completion status of "0b001 - Unsupported Request". When the slave bridge receives the unsupported request response, it discards the completion TLP and issues the Slave Unsupported Request (SUR) interrupt to the Bridge Interrupt Status Register.

Completion Timeout

A Completion Timeout occurs when a completion (Cpl) or completion with data (CplD) TLP is not returned after a PLB to PCIe request. The types of request that will cause a timeout are Memory read, IO read, IO write, Configuration read, and Configuration write. Completions must complete within the C_COMP_TIMEOUT parameter value of the PLB request. When a completion timeout occurs, the Slave Bridge issues a Slave Completion Timeout (SCT) interrupt to the Bridge Interrupt Status Register.

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 Bridge detects the poisoned packet, it discards the completion TLP and issues a Slave Error Poison (SEP) interrupt to the Bridge Interrupt Status Register.

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 Bridge receives the completer abort response, it discards the completion TLP and issues a Slave Completer Abort (SCA) interrupt to the Bridge Interrupt Status Register.

Configuration Read Response

The Configuration Read cycle generated via the ECAM is a special case for unsuccessful completions. Since the PCIe uses the methodology from PCI for discovery and enumeration, the bridge will respond on the PLB bus with data of all ones in the case of unsuccessful completions. Also, no interrupts are generated and no `PLB_MRdErr` is asserted.

Table 16: Response of PLBv46_PCIe bridge Slave Side to Abnormal TermInations

Transfer Type	Abnormal Condition	Bridge Response
Config Write via ECAM (RC Only)	Bus number greater than subordinate bus number and UR returned. (1)(2)(3)(4)(5)	Config Wr type 1 tlp sent. SUR interrupt asserted.
Config Write via ECAM (RC Only)	Bus number greater than subordinate bus number and completion timeout occurs. (1)(2)(3)(4)(5)	Config Wr type 1 tlp sent. SCT interrupt asserted.
Config Read via ECAM (RC Only)	Bus number greater than subordinate bus number and UR returned (1)(2)(4)	Config Rd type 1 tlp sent. No interrupt asserted. All ones returned on PLB bus.
Config Read via ECAM (RC Only)	Bus number greater than subordinate bus number and completion timeout occurs (1)(2)(4)	Config Rd type 1 tlp sent. No interrupt asserted. All ones returned on PLB bus.
IO Write (RC Only)	UR returned (1)(3)	SUR interrupt asserted.
IO Write (RC Only)	Completion timeout occurs (1)(3)	SCT interrupt asserted.
IO Read (RC Only)	UR returned (1)	SUR interrupt asserted. Arbitrary data returned on PLB bus. MRdErr asserted on PLB bus.
IO Read (RC Only)	Completion timeout occurs (1)	SCT interrupt asserted. Arbitrary data returned on PLB bus. MRdErr asserted on PLB bus.
Mem Read	UR returned (1)	SUR interrupt asserted. Arbitrary data returned on PLB bus. MRdErr asserted on PLB bus.
Mem Read	Completion timeout occurs (1)	SCT interrupt asserted. Arbitrary data returned on PLB bus. MRdErr asserted on PLB bus.

1. When non-posted requests are repeated, interrupts for all attempts are asserted (except for config Read).
2. Completion status of CRS will cause a retry of the configuration read or write request.
3. MWrErr was removed from the Bridge Response for Config Writes and IO Writes because the Slave IPIF does not support asserting MWrErr on singles when configured with buffers.
4. Note that SW should not read or write to a bus number greater than the value it sets in the subordinate register at enumeration time; however, we do specify the bridge behavior if this violation occurs. The bridge does not stop the SW from performing transactions not allowed by spec.
5. The behavior for Cfg writes is the same as IO writes. Because IO writes are non-posted and MemWr are posted, their behaviors are inherently different.

Master Side Abnormal Conditions

The Master Bridge will detect abnormal conditions by first checking for Invalid Requests (for example, message, I/O, configuration, other...). If the request is valid the Master Bridge will then check for a Bar Match. If the Bar Match is valid, then the Master Bridge checks the transaction layer packet header for the Error Poison bit. In the case the Master Bridge detects an invalid request, the Bridge will not check for Bar Match or Error Poison and finish processing the Invalid Request. The same is true in the case when the Master Bridge detects a invalid Bar Match. The Bridge will not check for Error Poison and finish processing the invalid Bar Match.

No BAR Match

This request is initiated by the remote requester and received by the PLBv46 Bridge. Because this request does not fall into one of the three PCI BARs, the request is discarded, then an Unsupported Request (UR) completion packet is sent if the request was Non Posted.

BAR Match – Invalid Request

This request is initiated by the remote requester and received by the PLBv46 Bridge and falls into one of the three PCI BARs. This request is not supported by the Master Bridge, such as I/O requests or address routed Messages, therefore the request is discarded and an Unsupported Request (UR) completion packet is sent if the request was Non Posted. In this case, an address routed Message is returned and the Master Unsupported Request (MUR) interrupt is set. If an I/O request is received, no Master Unsupported Request (MUR) interrupt is set.

PLB Master Timeout

When the PLB Master receives a PLB Timeout, the request is discarded and the Master Completer Abort (MCA) interrupt is issued to the Bridge Interrupt Status register. If the request was non-posted, the Master Bridge responds by sending a completion (Cpl) with the Completion Status = Completer Abort to the remote requester.

PLB Master Error

When a remote requester performs a read request from the PLBv46 Bridge and the addressed Slave on the PLB responds with a `PLB_MRdErr`, the Master Bridge responds by sending a completion (Cpl) with the Completion Status = Completer Abort to the remote requester.

Messages

All messages are terminated in the Bridge. When a Vendor Message is received, the response depends on the type. If a Type 1 Vendor message is received, the packet is discarded. If a Type 0 Vendor message is received, the message is discarded and the MUR interrupt is issued to the Bridge Interrupt Status register.

PCIe Max Payload Size, 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 4k 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 may result in the return of a completion that violates one of these conditions or the loss of data. The master bridge does not return a malformed TLP completion to signal this violation.

Completion Packets

A situation may exist where the PCIe read request may ask for more data than the Bridge can insert into a single completion packet. This situation can exist when the MAX_READ_REQUEST_SIZE is greater than the MAX_PAYLOAD_SIZE. When this situation occurs, multiple completion packets are generated up to the MAX_PAYLOAD_SIZE, except for the last one, which is the remainder of the request.

Poison Bit

When the poison bit is set in a transaction layer packet (TLP) header, the payload following the header is corrupt. When the Bridge receives a memory request TLP with the poison bit set, it will discard the TLP and issue the Master Error Poison (MEP) interrupt to the Bridge Interrupt Status register. When the Bridge receives a configuration request with the poison bit set, it returns a completion with Status = Unsupported Request.

Zero Length Requests

When the Bridge receives a read request from a remote requester with the Length = 0x1, FirstBE = 0x00, and LastBE = 0x00, it responds by sending a completion with Status = Successful Completion. When the Bridge receives a (non-posted) write request from a remote requester with the Length = 0x1, FirstBE = 0x00, and LastBE = 0x00, it has no effect.

TLP Byte Enables

The Bridge does not support non-contiguous byte enables in PCIe to PLB write request TLPs. When they are detected, the write request is discarded and the NBE interrupt is set in the Bridge Interrupt Status register. The Bridge is capable of supporting a read request from the Root Complex that has the allowed non-contiguous TLP byte enables.

Link Speed and Compliance

The link speed depends on the C_FAMILY and C_NO_OF_LANES parameters and the PCI Express Base Specification Compliance is dependent on the C_FAMILY parameter. Table 17 defines the link speed and PCI Express compliance based on Bridge configuration.

Table 17: **Configuration Link Speeds**

C_FAMILY	C_NO_OF_LANES	Link Speed	PCI Express Base Specification Compliance
spartan6	x1	2.5 Gbps	v1.1
virtex6	x1	2.5 Gbps	v2.0 [1]
virtex5	x1, x4, x8	2.5 Gbps	v1.1

1. The Virtex-6 FPGA PCIe is compliant with v2.0 of the PCIe specification, but the bridge does not support 5.0 Gbps link speed.

Limitations

Slave Bridge Limitations

Burst Size

The slave bridge supports bursts from masters that are smaller than the native size of the bridge. The maximum burst size supported for a 32-bit master is 64 bytes.

PLB Clock Frequency Range

The bridge has been verified in simulation to work with a PLB clock frequency range of 50 MHz to 150 MHz.

Constraints

The Bridge is asynchronous by design and care must taken to constrain all paths that cross the asynchronous clock domains. Once synthesis has finished, the user must copy the commented "Bridge clock domain crossing constraints" section from the core level user constraint file and paste this section into the system UCF. The core level constraint file is located in the bridge instance subdirectory in the EDK project implementation directory. Next, the user must uncomment the "Bridge clock domain crossing constraints" section and replace the <Add period constraint here> with the period value used by the `MPLB_Clk` and `SPLB_Clk` in the user system.mhs file. Once this is completed the user must run the EDK build bitstream for the new clock domain crossing constraints to be recognized by the tools.

Example Constraints

Example constraints for the ML505, ML555, ML507 and ML605 boards are provided in the following sections for reference. The ml505 and ml555 systems are asynchronous designs that use a 125 MHz system clock generated by a DCM that is connected to the `MPLB_Clk` and `SPLB_Clk` inputs to the core. The Bridge_Clk uses a 125 MHz clock generated from an internal PLL. The ml507 system is an asynchronous design that uses a 100 MHz system clock generated by a DCM that is connected to the `MPLB_Clk` and `SPLB_Clk` inputs to the core. The Bridge_Clk uses a 125 MHz clock generated from an internal PLL. The ml605 system is an asynchronous design that uses a 250 MHz system clock generated by a DCM that is connected to the `MPLB_Clk` and `SPLB_Clk` inputs to the core. The Bridge_Clk uses a 125 MHz clock generated from an internal PLL.

ML505 Constraints

```
#########################################################################################

# System level pin location constraints

#########################################################################################

Net system_clk_pin LOC=AH15; #100MHz

Net system_clk_pin IOSTANDARD = LVCMOS33;

Net system_reset_pin LOC=E9;

Net system_reset_pin IOSTANDARD=LVCMOS33;

Net perstn_pin LOC=W10;

Net perstn_pin IOSTANDARD=LVCMOS33;

Net ref_clk_p_pin<0> LOC=AF4;

Net ref_clk_n_pin<0> LOC=AF3;

#########################################################################################

# System level clock constraints

#########################################################################################

Net system_clk_pin TNM_NET = system_clk_pin;

TIMESPEC TS_system_clk_pin = PERIOD system_clk_pin 10 ns;

Net int_ref_clk TNM_NET = int_ref_clk;

TIMESPEC TS_int_ref_clk = PERIOD int_ref_clk 10 ns;
```

```
#######################################################################################
# PCIe TX/RX pin location constraints
#######################################################################################
Net RXN_pin<0> LOC=AF1;
Net RXN_pin<0> IOSTANDARD = LVDS_25;
Net RXP_pin<0> LOC=AE1;
Net RXP_pin<0> IOSTANDARD = LVDS_25;
Net TXN_pin<0> LOC=AE2;
Net TXN_pin<0> IOSTANDARD = LVDS_25;
Net TXP_pin<0> LOC=AD2;
Net TXP_pin<0> IOSTANDARD = LVDS_25;
########################################################################################
# Timing constraints
########################################################################################
 NET "plbv46_pcie_0/*core_clk" PERIOD =  4 ns;
 NET "plbv46_pcie_0/*Bridge_Clk" PERIOD =  8 ns;
#######################################################################################
# Physical Constraints
#######################################################################################
# Block RAM placement
INST "plbv46_pcie_0/*pcie_mim_wrapper_i/bram_tl_tx/generate_tdp2[1].ram_tdp2_inst"
LOC = RAMB36_X1Y9;
INST "plbv46_pcie_0/*pcie_mim_wrapper_i/bram_tl_rx/generate_tdp2[1].ram_tdp2_inst"
LOC = RAMB36_X1Y8;
INST "plbv46_pcie_0/*pcie_mim_wrapper_i/bram_tl_tx/generate_tdp2[0].ram_tdp2_inst"
LOC = RAMB36_X1Y7;
INST "plbv46_pcie_0/*pcie_mim_wrapper_i/bram_tl_rx/generate_tdp2[0].ram_tdp2_inst"
LOC = RAMB36_X1Y6;
INST "plbv46_pcie_0/*pcie_mim_wrapper_i/bram_retry/generate_sdp.ram_sdp_inst"
LOC = RAMB36_X1Y5;
# Timing critical placements
INST "plbv46_pcie_0/*tx_bridge/shift_pipe1” LOC = "SLICE_X59Y36";
INST "plbv46_pcie_0/*arb_inst/completion_available" LOC = "SLICE_X58Y26";
INST "plbv46_pcie_0/*management_interface/mgmt_rdata_d1_3” LOC = "SLICE_X59Y25";
#####################################################################################################
# Bridge clock domain crossing constraints
#####################################################################################################
 NET "plbv46_pcie_0/*SPLB_Clk"            TNM_NET = "SPLB_Clk";
 NET "plbv46_pcie_0/*Bridge_Clk"          TNM_NET = "Bridge_Clk";
 TIMESPEC "TS_PLB_PCIe" = FROM "SPLB_Clk"   TO "Bridge_Clk" 8 ns datapathonly;
 TIMESPEC "TS_PCIe_PLB" = FROM "Bridge_Clk" TO "SPLB_Clk" 10 ns datapathonly;
```

ML555 Constraints

```
####################################################################################################
# System level pin location constraints
####################################################################################################
Net system_clk_pin LOC=L19;# X1 = 33MHz
Net system_clk_pin IOSTANDARD = LVCMOS33;
Net system_reset_pin LOC=AF21;
Net system_reset_pin IOSTANDARD=LVCMOS33;
Net perstn_pin LOC=E14;
Net perstn_pin IOSTANDARD=LVCMOS33;
Net ref_clk_p_pin<0> LOC=Y4;
Net ref_clk_n_pin<0> LOC=Y3;
####################################################################################################
# System level clock constraints
####################################################################################################
Net system_clk_pin TNM_NET = system_clk_pin;
TIMESPEC TS_system_clk_pin = PERIOD system_clk_pin 10 ns;
Net int_ref_clk TNM_NET = int_ref_clk;
TIMESPEC TS_int_ref_clk = PERIOD int_ref_clk 10 ns;


####################################################################################################
# PCIe TX/RX pin location constraints
####################################################################################################
Net TXP_pin<0>         LOC =  V2;
Net TXP_pin<0> IOSTANDARD = LVDS_25;
Net TXN_pin<0>          LOC =  W2;
Net TXN_pin<0> IOSTANDARD = LVDS_25;
Net RXP_pin<0>          LOC =  W1;
Net RXP_pin<0> IOSTANDARD = LVDS_25;
Net RXN_pin<0>          LOC =  Y1;
Net TXP_pin<0> IOSTANDARD = LVDS_25;
########################################################################################
# Timing constraints
########################################################################################
 NET "plbv46_pcie_0/*core_clk" PERIOD =  4 ns;
 NET "plbv46_pcie_0/*Bridge_Clk" PERIOD =  8 ns;
#######################################################################################
# Physical Constraints
#######################################################################################
# BlockRAM placement
INST "plbv46_pcie_0/*pcie_mim_wrapper_i/bram_tl_tx/generate_tdp2[1].ram_tdp2_inst"
LOC = RAMB36_X1Y9;
INST "plbv46_pcie_0/*pcie_mim_wrapper_i/bram_tl_rx/generate_tdp2[1].ram_tdp2_inst"
LOC = RAMB36_X1Y8;
INST "plbv46_pcie_0/*pcie_mim_wrapper_i/bram_tl_tx/generate_tdp2[0].ram_tdp2_inst"
LOC = RAMB36_X1Y7;
INST "plbv46_pcie_0/*pcie_mim_wrapper_i/bram_tl_rx/generate_tdp2[0].ram_tdp2_inst"
LOC = RAMB36_X1Y6;
INST "plbv46_pcie_0/*pcie_mim_wrapper_i/bram_retry/generate_sdp.ram_sdp_inst"
LOC = RAMB36_X1Y5;
# Timing critical placements
INST "plbv46_pcie_0/*tx_bridge/shift_pipe1” LOC = "SLICE_X59Y36";
INST "plbv46_pcie_0/*arb_inst/completion_available" LOC = "SLICE_X58Y26";
INST "plbv46_pcie_0/*management_interface/mgmt_rdata_d1_3” LOC = "SLICE_X59Y25";
########################################################################################
# Bridge clock domain crossing constraints
########################################################################################
 NET "plbv46_pcie_0/*SPLB_Clk"          TNM_NET = "SPLB_Clk";
 NET "plbv46_pcie_0/*Bridge_Clk"          TNM_NET = "Bridge_Clk";
 TIMESPEC "TS_PLB_PCIe" = FROM "SPLB_Clk"   TO "Bridge_Clk" 8 ns datapathonly;
 TIMESPEC "TS_PCIe_PLB" = FROM "Bridge_Clk" TO "SPLB_Clk" 10 ns datapathonly;
```

ML507 Constraints

```
########################################################################################
# System level pin location constraints
########################################################################################
Net system_clk_pin LOC=AH15;# X1 = 100MHz
Net system_clk_pin IOSTANDARD = LVCMOS33;
Net system_reset_pin LOC=E9;
Net system_reset_pin IOSTANDARD=LVCMOS33;
Net perstn_pin LOC=W10;
Net perstn_pin IOSTANDARD=LVCMOS33;
Net ref_clk_p_pin<0> LOC=AF4;
Net ref_clk_n_pin<0> LOC=AF3;
########################################################################################
# System level clock constraints
########################################################################################
Net system_clk_pin TNM_NET = system_clk_pin;
TIMESPEC TS_system_clk_pin = PERIOD system_clk_pin 10 ns;
Net int_ref_clk TNM_NET = int_ref_clk;
TIMESPEC TS_int_ref_clk = PERIOD int_ref_clk 10 ns;
########################################################################################
# PCIe TX/RX pin location constraints
########################################################################################
Net RXN_pin<0> LOC=AF1;
Net RXN_pin<0> IOSTANDARD = LVDS_25;
Net RXP_pin<0> LOC=AE1;
Net RXP_pin<0> IOSTANDARD = LVDS_25;
Net TXN_pin<0> LOC=AE2;
Net TXN_pin<0> IOSTANDARD = LVDS_25;
Net TXP_pin<0> LOC=AD2;
Net TXP_pin<0> IOSTANDARD = LVDS_25;
```

```
##########################################################################################
# Timing constraints
##########################################################################################
Net "plbv46_pcie_0/*core_clk" PERIOD =  4 ns;
Net "plbv46_pcie_0/*Bridge_Clk" PERIOD =  8 ns;
#########################################################################################
# PCI Express Block placement
#########################################################################################
INST "plbv46_pcie_0/*pcie_ep"  LOC = PCIE_X0Y0;
#########################################################################################
# Physical Constraints
#########################################################################################
# BlockRAM placement
INST "plbv46_pcie_0/*pcie_blk/pcie_mim_wrapper_i/bram_retry/generate_sdp.ram_sdp_inst"
LOC = RAMB36_X4Y4;
INST "plbv46_pcie_0/*pcie_blk/pcie_mim_wrapper_i/bram_tl_tx/generate_tdp2[1].ram_tdp2_inst"
LOC = RAMB36_X4Y3;
INST "plbv46_pcie_0/*pcie_blk/pcie_mim_wrapper_i/bram_tl_rx/generate_tdp2[1].ram_tdp2_inst"
LOC = RAMB36_X4Y2;
INST "plbv46_pcie_0/*pcie_blk/pcie_mim_wrapper_i/bram_tl_tx/generate_tdp2[0].ram_tdp2_inst"
LOC = RAMB36_X4Y1;
INST "plbv46_pcie_0/*pcie_blk/pcie_mim_wrapper_i/bram_tl_rx/generate_tdp2[0].ram_tdp2_inst"
LOC = RAMB36_X4Y0;
# Timing critical placements
INST "plbv46_pcie_0/*tx_bridge/vld_q1"  LOC = "SLICE_X75Y16";
###########################################################################################
## Bridge clock domain crossing constraints
###########################################################################################
NET "plbv46_pcie_0/*MPLB_Clk" PERIOD =  10 ns;
NET "plbv46_pcie_0/*SPLB_Clk" PERIOD = 10 ns;
NET "plbv46_pcie_0/*SPLB_Clk" TNM_NET = "SPLB_Clk";
NET "plbv46_pcie_0/*Bridge_Clk" TNM_NET = "Bridge_Clk";
# Timing constraints between clock-domain boundaries
```

```
TIMESPEC "TS_PLB_PCIe" = FROM "SPLB_Clk" TO "Bridge_Clk" 8 ns datapathonly;
TIMESPEC "TS_PCIe_PLB" = FROM "Bridge_Clk" TO "SPLB_Clk" 10 ns datapathonly;
```

ML605 Constraints

```
###############################################################################
# System level pin location constraints
###############################################################################
Net fpga_0_clk_1_sys_clk_p_pin LOC = J9;
Net fpga_0_clk_1_sys_clk_p_pin IOSTANDARD = LVDS_25;
Net fpga_0_clk_1_sys_clk_p_pin DIFF_TERM = TRUE;
Net fpga_0_clk_1_sys_clk_n_pin LOC = H9;
Net fpga_0_clk_1_sys_clk_n_pin IOSTANDARD = LVDS_25;
Net fpga_0_clk_1_sys_clk_n_pin DIFF_TERM = TRUE;
Net fpga_0_rst_1_sys_rst_pin LOC = H10;
Net fpga_0_rst_1_sys_rst_pin IOSTANDARD = SSTL15;
Net fpga_0_rst_1_sys_rst_pin PULLUP;
Net fpga_0_rst_1_sys_rst_pin TIG;
###############################################################################
# System level clock constraints
###############################################################################
NET "PCIe_Bridge/REFCLK" TNM_NET = "PCIe_RefClk" ;
NET "*/pcie_clocking_i/clk_125" TNM_NET = "PCIe_CLK_125" ;
TIMESPEC "TS_PCIe_RefClk"  = PERIOD "PCIe_RefClk" 250.00 MHz HIGH 50 % ;
 TIMESPEC "TS_PCIe_CLK_125" = PERIOD "PCIe_CLK_125" TS_PCIe_RefClk/2.0 HIGH 50 % PRIORITY 100;
###############################################################################
###### PCIe_Bridge constraints
###############################################################################
# SYS clock 250 MHz (input) signal.  The sys_clk_p and sys_clk_n
# signals are the PCI Express reference clock. Virtex-6 FPGA GT
# Transceiver architecture requires the use of a dedicated clock
# resources (FPGA input pins) associated with each GT Transceiver.
# To use these pins an IBUFDS primitive (refclk_ibuf) is
# instantiated in user's design.
# See the Virtex-5 FPGA GT Transceiver User Guide
# (UG) for guidelines regarding clock resource selection.
INST     "*/PCIe_Diff_Clk/USE_IBUFDS_GTXE1.GEN_IBUFDS_GTXE1[0].IBUFDS_GTXE1_I"     LOC     =
IBUFDS_GTXE1_X0Y4;
# Transceiver instance placement. This constraint selects the
# transceivers to be used, which also dictates the pinout for the
# transmit and receive differential pairs. See the # Virtex-6 FPGA GT Transceiver User Guide (UG) for more
# information.
# PCIe Lane 0
INST "*/pcie_2_0_i/pcie_gt_i/gtx_v6_i/GTXD[0].GTX" LOC = GTXE1_X0Y15;
# PCI Express Block placement. This constraint selects the PCI Express
# Block to be used. #
 INST "*/pcie_2_0_i/pcie_block_i" LOC = PCIE_X0Y1;
################################################################################
# Bridge clock domain crossing constraints
################################################################################
 NET "plbv46_pcie_0/*SPLB_Clk"          TNM_NET = "SPLB_Clk";
 NET "plbv46_pcie_0/*Bridge_Clk"        TNM_NET = "Bridge_Clk";
 TIMESPEC "TS_PLB_PCIe" = FROM "SPLB_Clk"   TO "Bridge_Clk" 8 ns datapathonly;
 TIMESPEC "TS_PCIe_PLB" = FROM "Bridge_Clk" TO "SPLB_Clk" 10 ns datapathonly;
```

ModelSim Simulation Notes

When simulating with ModelSim, the vsim options -L XilinxCoreLib_ver, -L secureip and -L unisims_ver must be set.

Timing

When building a system in Virtex-5 FPGA hardware, set the “-xe c” and “-ol high” switches for both MAP and PAR so that the core will meet timing.

Device Utilization and Performance Benchmarks

System Performance

To measure the performance (F_{MAX}) of the PLBv46 Endpoint Bridge core, it was added to the Virtex-5 FPGA system shown in Figure 8.

Because the PLBv46 Bridge core will be used with other design modules in the FPGA, the utilization and timing numbers reported in this section are estimates only. When this core is combined with other design modules in the system, the utilization of FPGA resources and timing of the core design will vary from the results reported here.

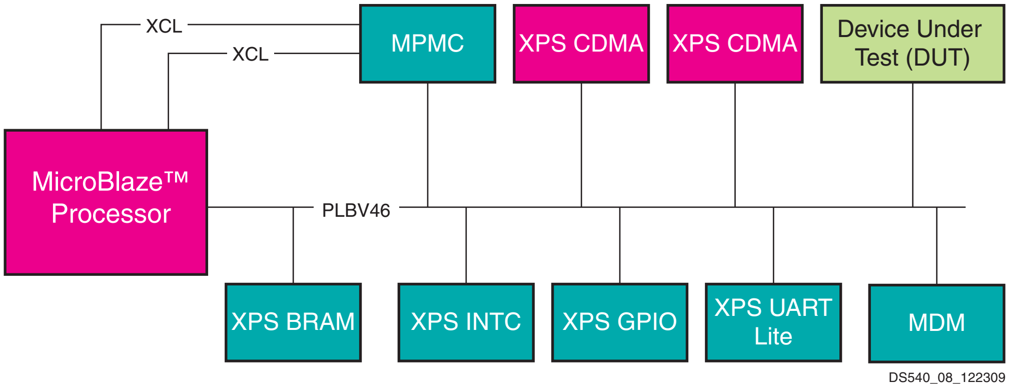

Figure 8: **Performance of the Virtex-5 FPGA System Using the PLBv46 Endpoint as the DUT**

Subsequently, the target FPGA listed in Table 18 is filled with logic to drive the LUT and block RAM utilization to approximately 70% and the I/O utilization to approximately 80%. Using the default tool options and the slowest speed grade for the target FPGA, the resulting target F_{MAX} numbers are shown in Table 18.

Table 18: **XPS PLBv46 Endpoint Core System Performance**

Target FPGA	Target F_{MAX} (MHz)
V5LXT50 -1	120

The target F_{MAX} is influenced by the exact system and is provided for guidance only. It is not a guaranteed value across all systems.

Resource Utilization

The PLBv46 Bridge core resources used in the Virtex-5, Virtex-6, or Spartan-6 FPGAs are detailed in Table 19.

Table 19: **PLBv46 Bridge Core Resources Used**

Device Family	Resource	Max	Min
Virtex-5	Slice LUT	8231[1]	7833[2]
	Slice Register	7497[1]	6871[2]
	BRAM36	10[1]	10[2]
	BRAM18	6[1]	6 [2]
Virtex -6	Slice LUT	3762[3]	3330[2]
	Slice Register	2505[3]	2130[2]
	BRAM36	6[3]	6[2]
	BRAM18	2[3]	2[2]
Spartan-6	Slice LUT	2868[3]	2658[2]
	Slice Register	2208[3]	1833[2]
	BRAM16	9[3]	9[2]
	BRAM8	2[3]	2[2]

1. 6 IPIFBAR (64-bit), 3 PCIBAR (64-bit) and x8 lane width
2. 1 IPIFBAR (32-bit), 1 PCIBAR (64-bit) and x1 lane width
3. 6 IPIFBAR (32-bit), 3 PCIBAR (64-bit) and x1 lane width

Specification Exceptions

N/A

Reference Documents

1. UG341 *LogiCORE IP Endpoint Block Plus v1.13 for PCI Express User Guide*
2. UG654 *LogiCORE IP Spartan-6 FPGA Integrated Endpoint Block v1.2 for PCI Express User Guide*
3. UG517 *LogiCORE IP Virtex-6 FPGA Integrated Block v1.4 for PCI Express User Guide*
4. DS566 *PLBV46 Master*
5. 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 Information

This Xilinx LogiCORE IP module is provided under the terms of the Xilinx Core Site License. The core is generated using the Xilinx ISE® Embedded Edition software (EDK). For full access to all core functionality in simulation and in hardware, you must purchase a license for the core. Please contact your local Xilinx sales representative for information on pricing and availability of Xilinx LogiCORE IP

For more information, please visit the PLBv46 Bridge for PCIe Express product web page.

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 IP modules and software, please contact your local Xilinx sales representative.

This Embedded IP module is provided under the terms of the Xilinx Core Site License. A free evaluation version of the core is included with the ISE Design Suite Embedded Edition software. Use the Xilinx Platform Studio application (XPS) included with the Embedded Edition software to instantiate and use this core.

For full access to all core functionality in simulation and in hardware, you must purchase a license for the core. Please contact your local Xilinx sales representative for information on pricing and availability of Xilinx LogiCORE IP.

List of Acronyms

Acronym	Spelled Out
BAR	Base Address Register
BCR	Bridge Control Register
BIER	Bridge Interrupt Enable Register
BIR	Bridge Interrupt Register
Cpl	Completion
CplD	Completion with Data
DCM	Digital Clock Manager
DUT	Device Under Test
ECAM	Enhanced Configuration Access Mechanism
EDK	Embedded Development Kit
EP	Endpoint
FIFO	First In First Out
FPGA	Field Programmable Gate Array
GT	GigaTransfers
I/O	Input/Output
IP	Intellectual Property
IPIF	IP Interface
ISE	Integrated Software Environment
LUT	Lookup Table
MCA	Master Completer Abort
MEP	Master Error Poison
MHz	MegaHertz

Acronym	Spelled Out
MPLB	Master PLB
MUR	Master Unsupported Request
PLB	Processor Local Bus
PLL	Phase-Locked Loop
PRCR	PCIe Requester Control Register
PRIDR	PCIe Requester ID Register
PSR	PCIe Status Register
R	Read
RAM	Random Access Memory
RC	Root Complex
RO	Read Only
SBO	Slave BAR Overrun
SCA	Slave Completer Abort
SCT	Slave Completion Timeout
SEP	Slave Error Poison
SPLB	Slave PLB
SW	Software
TLIF	Transaction Layer Interface
TLP	Transaction Layer Packets
UCF	User Constraints File
UR	Unsupported Request
VHDL	VHSIC Hardware Description Language (VHSIC an acronym for Very High-Speed Integrated Circuits)
XPS	Xilinx Platform Studio
XST	Xilinx Synthesis Technology

Revision History

Date	Version	Revision
8/28/08	1.0	Initial Xilinx Release (v3.00a)
9/10/08	1.1	Updated version to v3.00b, EDK_11.1; updated margin system F_{MAX} requirements; see change log for specific changes.
4/24/09	1.2	Replaced device families and tools with hyperlink.
6/24/09	1.3	Updated to v4.01a, EDK 11.2 release; added Virtex-6 and Spartan-6 support (including lane support and respective parameters for same); see change log for specific changes.
9/16/09	1.4	Updated to v4.02a for EDK 11.3 release; added Root Complex Capability for Virtex 6
12/2/09	1.5	Created v4.03a for EDK 11.4 release; converted to current data sheet template.
4/19/10	1.6	Created v4.04a for EDK 12.1 release.
9/21/10	1.7	Created v4.05a for EDK 12.3 release.
12/14/10	1.8	Created v4.06.a for EDK 12.4 release.

Notice of Disclaimer

Xilinx is providing this product documentation, hereinafter "Information," to you "AS IS" with no warranty of any kind, express or implied. Xilinx makes no representation that the Information, or any particular implementation thereof, is free from any claims of infringement. You are responsible for obtaining any rights you may require for any implementation based on the Information. All specifications are subject to change without notice. XILINX EXPRESSLY DISCLAIMS ANY WARRANTY WHATSOEVER WITH RESPECT TO THE ADEQUACY OF THE INFORMATION OR ANY IMPLEMENTATION BASED THEREON, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OR REPRESENTATIONS THAT THIS IMPLEMENTATION IS FREE FROM CLAIMS OF INFRINGEMENT AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Except as stated herein, none of the Information may be copied, reproduced, distributed, republished, downloaded, displayed, posted, or transmitted in any form or by any means including, but not limited to, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Xilinx.