LogiCORE IP PLBV46 PCI Full Bridge (v1.04.a)

DS616 June 22, 2011 **Product Specification**

Introduction

The PLBV46 PCI™ Full Bridge design provides full bridge functionality between the Xilinx® PLB and a 32-bit Revision 2.2 compliant Peripheral Component Interconnect (PCI™) bus. The bridge is referred to as the PLBV46 PCI Bridge in this document.

The Xilinx PLB is a 32, 64 or 128-bit bus subset of the IBM PLB described in the *128-Bit Processor Local Bus Architecture Specification v4.6*.

The LogiCORE™ IP PCI32 core provides an interface with the PCI bus. Details of the LogiCORE IP PCI32 core operation is found in the *Xilinx LogiCORE PCI32 Interface v3*, in the *Xilinx LogiCORE PCI32 Interface v4 Product Specification*, and in the *Xilinx LogiCORE PCI v3.0 and v4.1 User Guides*.

Host bridge functionality (often called North bridge functionality) is an optional functionality. Configuration Read and Write PCI commands can be performed from the PLB-side of the bridge. The PLBV46 PCI Bridge supports a 32-bit/33 MHz PCI bus only.

Exceptions to the support of PCI commands supported by the PCI32 core are outlined in the Features section.

The PLBV46 PCI Bridge design has parameters that allow customers to configure the bridge to suit their application. The parameterizable features of the design are discussed in the Bus Interface Parameters section.

LogiCORE IP Facts Table					
Core Specifics					
Supported Device Family[1]	Virtex-4, Virtex-5 Spartan-3, Spartan-6				
Supported User Interfaces	plbv46, pci				
Resources					
	LUTs	**FFs**	**I/O (PCI)**	**I/O (PLB related)**	**Block RAMs**
	See Table 29.				
Provided with Core					
Documentation	Product Specification				
Design Files	VHDL				
Example Design	Not Provided				
Test Bench	Not Provided				
Constraints File	example UCF-file				
Simulation Model	Not Provided				
Tested Design Tools					
Design Entry Tools	ISE v13.2 software				
Simulation	Mentor Graphics ModelSim [2]				
Synthesis Tools	XST 13.2				
Support					
Provided by Xilinx, Inc.					

1. For a complete listing of supported devices, see IDS Embedded Edition Derivative Device Support for this core.
2. For the supported versions of the tools, see the ISE Design Suite 13: Release Notes Guide.

Features

- Independent SPLB, MPLB and PCI clocks
- 33 MHz, 32-bit PCI bus support
- Utilizes two pairs of FIFOs to exploit the separate master and slave PLBV46 IPIF modules.
- Includes a master IP module for remote PCI initiator transactions, which follows the protocol for interfacing with the master IPIF module utilizing Xilinx LocalLink protocol. The PLBV46 PCI Bridge translates the PCI initiator request to PLBV46 IPIF master transactions.
- Includes a slave IP module for remote PLB master transactions, which follows the protocol for interfacing with the slave IPIF module utilizing Xilinx IPIC protocol. The PLBV46 PCI Bridge translates the PLB master request to PCI initiator transactions. The SRAM-like interface is utilized at the IPIC interface for data transfers.
- The PLBV46 IPIF slave attachment has a timer that limits the time for both read and write data phase operations to complete. When the timer expires, Sl_MErr signal is asserted. See the PLBV46 IPIF Product Specification for details.
- Full bridge functionality
  - PLB Master read and write of a remote PCI target (both single and burst)
  - PCI Initiator read and write to a remote PLB slave (both single and multiple).
  - I/O read and I/O write commands are supported only for PLB master read and writes of PCI I/O space as designated by its associated memory designator parameter. All memory space on the PLB-side is designated as memory space in the PCI sense, therefore, I/O commands cannot be used to access memory on the PLB-side.
  - Configuration read and writes are supported (including self-configuration transactions) only when upper word address lines are utilized for IDSEL lines. The Configuration Read and Write commands are automatically executed by writing to the Configuration Data Port Register. Data in the Configuration Address Port Register and the Configuration Bus Number/Subordinate Bus Number Register are used in execution of the configuration transaction per PCI 2.2 specification.
- PCI Memory Read Line (MRL) command is supported in which the PCI32 core is a target. MRL is aliased to a Memory Read command which has a single data phase on the PCI.
- PCI Memory Write Invalidate (MWI) command is supported in which the PCI32 core is a target. The PCI32 core does not support this command when it is an initiator. MWI is aliased to a Memory Write command which has a single data phase on the PCI.
- Supports up to 6 PLB devices, in the sense defined by independent parameters and unique PLB memory space for each device
  - Each device has the following parameters: PLB BAR, high (upper) address, memory designator, and translation for mapping PLB address space to PCI address space. Byte addressing integrity is maintained by default in all transfers. Address translation is performed by high-order bit substitution. High-order bit definition can be done with parameters or dynamically via registers.
- Supports up to 3 PCI devices (or BARs in PCI context) with unique memory PCI memory space. The PCI32 core supports up to 3 PCI BAR.
  - Each device has the following parameters: PCI BAR, length, memory designator, and translation for mapping PCI address space to PLB address space. Byte addressing integrity is maintained by default in all transfers. Address translation is performed by high-order bit substitution. High-order bit definition is defined only by parameters

- Registers include
  - Interrupt and interrupt enable registers at different hierarchal levels
  - Reset
  - Configuration Address Port, Configuration Data Port and Bus Number/Subordinate Bus Number
  - High-order bits for PLB to PCI address translation
  - Bridge Device number on PCI bus
- PLB-side Interrupts include
  - PLB Master Read SERR and PERR
  - PLB Master Read Target Abort
  - PLB Master Write SERR and PERR
  - PLB Master Write Target Abort
  - PLB Master Write Master Abort
  - PLB Master Burst Write Retry and Retry Disconnect
  - PLB Master Burst Write Retry Timeout
  - PCI Initiator Read and Write SERR
  - PLB Master Prefetch Timeout
  - PLB Master Write Rearb Timeout
  - PLB Master Read Rearb Timeout
- Asynchronous FIFOs with burst transfer support and backup capability for retrying transfers as needed. The maximum burst size on either the PCI or PLB is limited to the usable FIFO depth which is the physical depth-3
- Synchronization circuits for signals that cross time-domain boundaries
- Responds to the PCI latency timer
- Completes posted write operations prior to initiating new operations
- Signal set required for integrating a PCI bus arbiter in the FPGA with the PLBV46 PCI Bridge is available at the top-level of the PLBV46 PCI Bridge module. The signal set includes PCLK, RST_N, FRAME_I, REQ_N_toArb and IRDY_I
- Supports PCI clock generated in FPGA
- Parameterized control of I/O-buffer insertion of INTR_A and REQ_N IO-buffers
- All address translations performed by high-order bit substitution. The number of bits substituted depends on the address range
  - Parameterized selection of IPIF BAR high-order bits defined by programmable registers for dynamic translation operation or by parameters for reduced resource utilization
- Parameterized selection of device ID number (when configuration functionality is included) defined by a programmable register for dynamic device number definition or by parameter to reduce resource utilization
- The PLBV46 PCI Bridge does not have an integral DMA
- Input signal to provide the means to asynchronous assert INTR_A from a user supplied register. such as the PLB GPIO register. The signal is Bus2PCI_INTR is an active high signal
- PCI Monitor output port to monitor PCI bus activity

System Reset

When the bridge is reset, both RST_N and PLB_reset must be simultaneously held at *reset* for at least twenty clock periods of the slowest clock.

Evaluation Version

The PLBV46 PCI Bridge is delivered with a hardware evaluation license. When programmed into a Xilinx device, the core will function in hardware for about 8 hours at the typical frequency of operation. To use the PLBV46 PCI Bridge without this timeout limitation, a full license must be purchased.

Functional Description

The PLBV46 PCI Bridge design is shown in Figure 1 and described in the following sections. As shown, PLB IPIF PCI Bridge is comprised of three main modules:

- The PLB IPIF (Processor Local Bus Intellectual Property InterFace). It interfaces to the PLB bus.
- The IPIF v3.0 Bridge. It interfaces between the PLBV46 IPIF and the PCI32 core.
- The LogiCORE IP PCI32 core. It interfaces to the PCI bus.

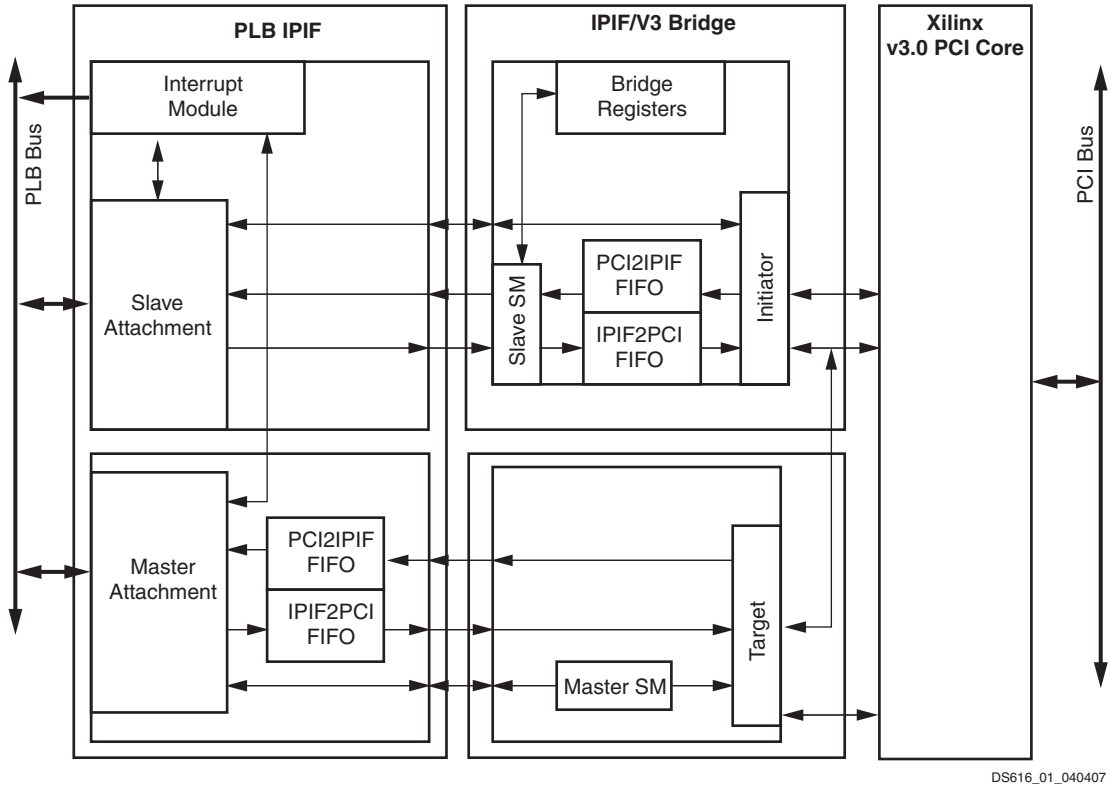

Figure 1: **PLBV46 PCI Full Bridge Block Diagram**

LogiCore IP 32-Bit PCI Core Requirements

The PLBV46 PCI Bridge uses the 32-bit PCI32 core. Before the bridge can perform transactions on the PCI bus, the PCI32 core must be configured via configuration transactions from either the PCI-side or if configuration functionality is included in the bridge configuration, from the PLB-side. Both a design guide and an implementation guide are available for the PCI32 IP core. These documents detail the PCI32 core operation, including configuration cycles, and are available from Xilinx.

The PCI32 core documents and PCI Specification contain design requirements that must be followed but are beyond the scope of this document. One example is that, according to the PCI specification, pull-up resistors are required on the system board for all PCI control signals. See the Reference Documents section for links to these documents.

As required by the PCI32 core, GNT_N must be asserted for two clock cycles to initiate a PCI transaction by the PLBV46 PCI Bridge.

Bus Interface Parameters

Because many features in the IPIF PCI Bridge design can be parameterized, the user can realize a PLBV46 PCI Full 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. Table 1 shown the features that can be parameterized in the PLBV46 PCI Bridge design.

Address Translation

Address space on the PCI side that is accessible from the PLB side must be translated to a 2^N contiguous block on the PLB side. Up to six contiguous blocks are possible. Each block has parameters for base address (C_IPIFBAR_N), high address, address translation vector, and memory designator (memory or I/O).

All address space on the PLB side that is accessible from the PCI side must be translated to a maximum of three 2^N contiguous blocks on the PCI side. Up to three blocks are possible because the PCI32 core supports up to 3 BARs. Each block has parameters for length, which must be a 2^N range, and address translation vector. Only PCI memory space is supported.

Address translations in both directions are performed as follows:

- High-order address bits are substituted for the address vector before crossing to the other bus domain. The number of high-order bits substituted in the PLB address presented to the bridge is given by the number of bits that are the same between the C_IPIFBAR_N and C_ IPIF_HIGHADDR_N parameters. The number of high-order bits substituted in the PCI address presented to the bridge for a translation from PCI to PLB domains is given by the bus width minus the parameter C_PCIBAR_LEN_N.
- The low-order bits are transferred directly between bus domains. The bits substituted in a translation from PLB to PCI domains can be selected via a parameter (C_INCLUDE_BAROFFSET_REG) as either a parameter (C_IPIFBAR2PCIBAR_N) or a programmable register for each BAR. The bits that are substituted for in a translation from PCI to PLB domains is defined by a parameter (C_PCIBAR2IPIFBAR_M) for each BAR.

Figure 2 shows two sets of base address register (BAR) parameters and how they are used. The two sets are independent sets: one set for the up to six PLB-side device (IPIFBAR) address ranges and another set for the up to three PCI-side device (PCIBAR) address ranges.

This document includes three examples of how to use the two sets of base address register (BAR) parameters:

Example 1, shown in Figure 2, outlines the use of the two sets of BAR parameters.

Example 2 outlines the use of the IPIFBAR parameters sets for the specific address translations of PLB addresses within the range of a given *IPIFBAR to a remote PCI address space*.

Example 3 outlines the use of the PCIBAR parameter sets for the address translation of PCI addresses within the range of a given *PCIBAR to a remote PLB address space*.

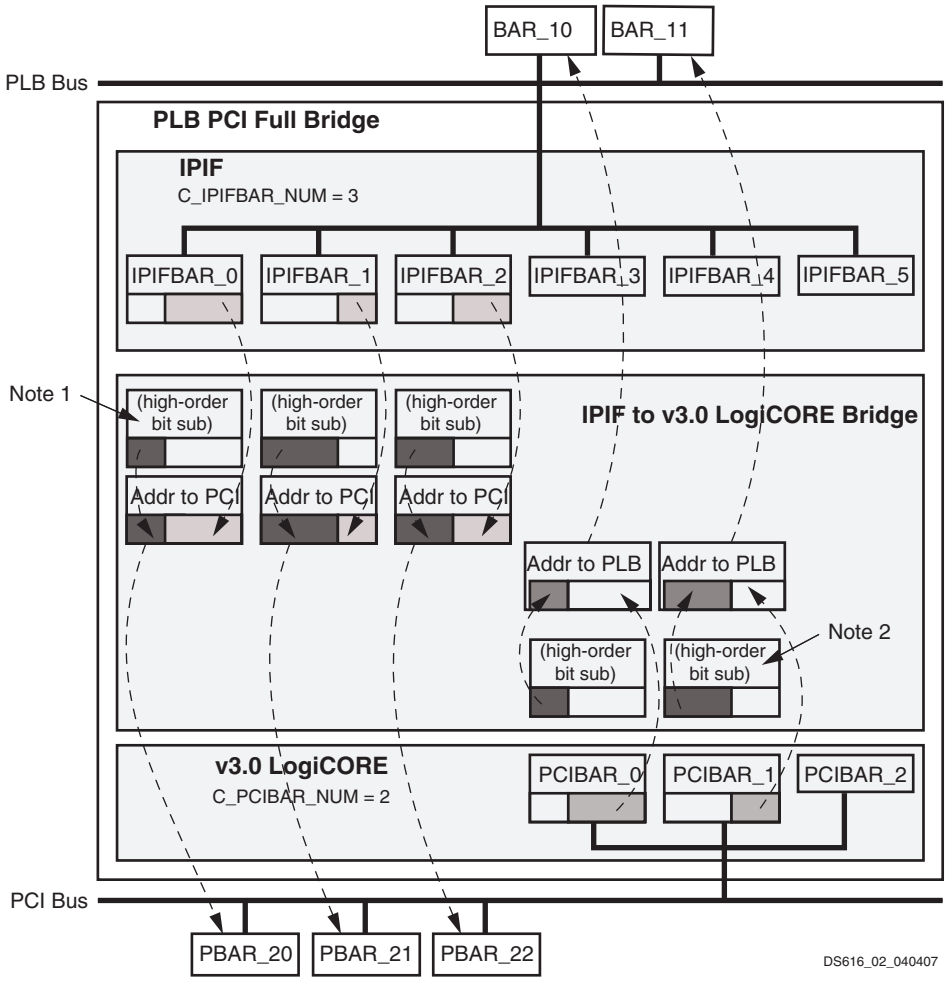

Figure 2: **Translation of Addresses Bus-to-Bus with High-Order Bit Substitution**

Example 1

Because address translations are performed only when the PLBV46 PCI Bridge is configured with FIFOs, the example shown in Figure 2 is for an *PLBV46 PCI Bridge configuration with FIFOs only*. In this example, it is assumed that C_INCLUDE_BAROFFSET_REG=0, therefore, the parameters C_IPIFBAR2PCIBAR_N define the high-order bits for substitution in translating the address on the PLB bus to the PCI bus.

The PLB parameters are C_IPIFBAR_N, C_IPIF_HIGHADDR_N, and C_IPIFBAR2PCIBAR_N for N=0 to 5.

The PCI parameters are C_PCIBAR_LEN_M and C_PCIBAR2IPIFBAR_M for M=0 to 2.

Example 2

Example 2 shows of the settings of the two independent sets of base address register (BAR) parameters for specifics of address translation of PLB addresses within the range of a given IPIFBAR to a remote PCI address space. This setting does not depend on the PCIBARs of the PLBV46 PCI Bridge.

As in example 1, it is assumed that the parameter C_INCLUDE_BAROFFSET_REG=0, therefore the C_IPIFBAR2PCIBAR_N parameters define the address translation.

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

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

C_IPIFBAR_1=0xABCDE000
C_IPIF_HIGHADDR_1=0xABCDFFFF
C_IPIFBAR2PCIBAR_1=0xFEDC0xXX (Bits 19-31 are don't cares)

C_IPIFBAR_2=0xFE000000
C_IPIF_HIGHADDR_2=0xFFFFFFFF
C_IPIFBAR2PCIBAR_2=0x40xXXXXX (Bits 7-31 are don't cares)

C_IPIFBAR_3=0x00000000
C_IPIF_HIGHADDR_3=0x0000007F
C_IPIFBAR2PCIBAR_3=8765438X (Bits 25-31 are don't cares)
```

Accessing the PLBV46 PCI Bridge IPIFBAR_0 with address `0x12340ABC` on the PLB bus yields `0x56710ABC` on the PCI bus.

Accessing the PLBV46 PCI Bridge IPIFBAR_1 with address `0xABCDF123` on the PLB bus yields `0xFEDC1123` on the PCI bus.

Accessing the PLBV46 PCI Bridge IPIFBAR_2 with address `0xFFFEDCBA` on the PLB bus yields `0x41FEDCBA` on the PCI bus.

Accessing the PLBV46 PCI Bridge IPIFBAR_3 with address `0x00000071` on the PLB bus yields `Ox876543F1` on the PCI bus.

Example 3

Example 3 outlines address translation of PCI addresses within the range of a given PCIBAR to PLB address space. This translation is independent of the PLBV46 PCI Bridge IPIF BARs.

The parameters C_PCIBAR2IPIFBAR_M parameters define the address translation for all C_PCIBAR_NUM.

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

```
BAR 0 is set to 0xABCDE800 by host
C_PCIBAR_LEN_0=11
C_PCIBAR2IPIFBAR_0=0x123450XX (Bits 21-31 are don't cares)

BAR 1 is set to 0x12000000 by host
C_PCIBAR_LEN_1=25
C_PCIBAR2IPIFBAR_1=0xFEXXXXXX (Bits 7-31 are don't cares)
```

Accessing the PLBV46 PCI Bridge PCIBAR_0 with address `0xABCDEFF4` on the PCI bus yields `0x123457F4` on the PLB bus.

Accessing the PLBV46 PCI Bridge PCIBAR_1 with address `0x1235FEDC` on the PCI bus yields `0xFE35FEDC` on the PLB bus.

Table 1: **PLBV46 PCI Bridge Interface Design Parameters**

Generic	Feature / Description	Parameter Name	Allowable Values	Default Value	VHDL Type
Bridge Features Parameter Group					
G1	Number of IPIF devices	C_IPIFBAR_NUM	1-6; Parameters listed in this table corresponding to unused BARs are ignored, but must be valid values. BAR label 0 is the required bar for all values 1-6 and the index increments from 0 as BARs are added	6	integer
G2	IPIF device 0 BAR	C_IPIFBAR_0	Valid PLB address (1), (4)	0xFFFFFFFF	std_logic_vector
G3	IPIF BAR high address 0	C_IPIF_HIGHADDR_0	Valid PLB address (1), (4)	0x00000000	std_logic_vector
G4	PCI BAR to which IPIF BAR 0 is mapped unless C_INCLUDE_BAROFFSET_REG = 1	C_IPIFBAR2PCIBAR_0	Vector of length C_SPLB_AWIDTH	0xFFFFFFFF	std_logic_vector
G5	IPIF BAR 0 memory designator	C_IPIF_SPACETYPE_0	0 = I/O space 1 = Memory space	1	integer
G6	IPIF device 1 BAR	C_IPIFBAR_1	Valid PLB address (1), (4)	0xFFFFFFFF	std_logic_vector
G7	IPIF BAR high address 1	C_IPIF_HIGHADDR_1	Valid PLB address (1), (4)	0x00000000	std_logic_vector
G8	PCI BAR to which IPIF BAR 1 is mapped unless C_INCLUDE_BAROFFSET_REG = 1	C_IPIFBAR2PCIBAR_1	Vector of length C_SPLB_AWIDTH	0xFFFFFFFF	std_logic_vector
G9	IPIF BAR 1 memory designator	C_IPIF_SPACETYPE_1	0 = I/O space 1 = Memory space	1	integer
G10	IPIF device 2 BAR	C_IPIFBAR_2	Valid PLB address (1), (4)	0xFFFFFFFF	std_logic_vector
G11	IPIF BAR high address 2	C_IPIF_HIGHADDR_2	Valid PLB address (1), (4)	0x00000000	std_logic_vector
G12	PCI BAR to which IPIF BAR 2 is mapped unless C_INCLUDE_BAROFFSET_REG = 1	C_IPIFBAR2PCIBAR_2	Vector of length C_SPLB_AWIDTH	0xFFFFFFFF	std_logic_vector
G13	IPIF BAR 2 memory designator	C_IPIF_SPACETYPE_2	0 = I/O space 1 = Memory space	1	integer
G14	IPIF device 3 BAR	C_IPIFBAR_3	Valid PLB address (1), (4)	0xFFFFFFFF	std_logic_vector
G15	IPIF BAR high address 3	C_IPIF_HIGHADDR_3	Valid PLB address (1), (4)	0x00000000	std_logic_vector
G16	PCI BAR to which IPIF BAR 3 is mapped unless C_INCLUDE_BAROFFSET_REG = 1.	C_IPIFBAR2PCIBAR_3	Vector of length C_SPLB_AWIDTH	0xFFFFFFFF	std_logic_vector

Table 1: **PLBV46 PCI Bridge Interface Design Parameters *(Cont'd)***

Generic	Feature / Description	Parameter Name	Allowable Values	Default Value	VHDL Type
G17	IPIF BAR 3 memory designator	C_IPIF_ SPACETYPE_3	0 = I/O space 1 = Memory space	1	integer
G18	IPIF device 4 BAR	C_IPIFBAR_4	Valid PLB address (1), (4)	0xFFFFFFFF	std_logic_ vector
G19	IPIF BAR high address 4	C_IPIF_ HIGHADDR_4	Valid PLB address (1), (4)	0x00000000	std_logic_ vector
G20	PCI BAR to which IPIF BAR 4 is mapped unless C_INCLUDE_BAROFFSET _REG = 1	C_IPIFBAR2 PCIBAR_4	Vector of length C_SPLB_AWIDTH	0xFFFFFFFF	std_logic_ vector
G21	IPIF BAR 4 memory designator	C_IPIF_SPACE TYPE_4	0 = I/O space 1 = Memory space	1	integer
G22	IPIF device 5 BAR	C_IPIFBAR_5	Valid PLB address (1), (4)	0xFFFFFFFF	std_logic_ vector
G23	IPIF BAR high address 5	C_IPIF_ HIGHADDR_5	Valid PLB address (1), (4)	0x00000000	std_logic_ve ctor
G24	PCI BAR to which IPIF BAR 5 is mapped unless C_INCLUDE_BAROFFSET _REG = 1	C_IPIFBAR2 PCIBAR_5	Vector of length C_SPLB_AWIDTH	0xFFFFFFFF	std_logic_ vector
G25	IPIF BAR 5 memory designator	C_IPIF_SPACE TYPE_5	0 = I/O space 1 = Memory space	1	integer
G26	Number of PCI devices	C_PCIBAR_ NUM	1-3; Parameters listed in the following rows corresponding to unused BARs are ignored, but must be valid values. BAR label 0 is the required bar for all values 1-3 and the index increments from 0 as BARs are added.	3	integer
G27	IPIF BAR to which PCI BAR 0 is mapped	C_PCIBAR2 IPIFBAR_0	Vector of length C_MPLB_AWIDTH	0x00000000	std_logic_ vector
G28	Power of 2 in the size in bytes of PCI BAR 0 space	C_PCIBAR_ LEN_0	5 to 30	16	integer
G29	IPIF BAR to which PCI BAR 1 is mapped	C_PCIBAR2IPIFB AR_1	Vector of length C_MPLB_AWIDTH	0x00000000	std_logic_ vector
G30	Power of 2 in the size in bytes of PCI BAR 1 space	C_PCIBAR_ LEN_1	5 to 30	16	integer
G31	IPIF BAR to which PCI BAR 2 is mapped	C_PCIBAR2 IPIFBAR_2	Vector of length C_MPLB_AWIDTH	0x00000000	std_logic_ vector
G32	Power of 2 in the size in bytes of PCI BAR 2 space	C_PCIBAR_ LEN_2	5 to 30	16	integer
G33	PCI address bus width	C_PCI_ABUS_ WIDTH	32	32	integer
G34	PCI data bus width	C_PCI_DBUS_ WIDTH	32	32	integer

Table 1: **PLBV46 PCI Bridge Interface Design Parameters *(Cont'd)***

Generic	Feature / Description	Parameter Name	Allowable Values	Default Value	VHDL Type
G35	Both PCI2IPIF FIFO address bus widths. Usable depth is 2^C_PCI2IPIF_FIFO_ABUS_WIDTH - 3	C_PCI2IPIF_FIFO_ABUS_WIDTH	7-11(3)	9	integer
G36	Both IPIF2PCI FIFO address bus widths. Usable depth is 2^C_IPIF2PCI_FIFO_ABUS_WIDTH - 3	C_IPIF2PCI_FIFO_ABUS_WIDTH	7-11(3)	9	integer
G37	Include explicit instantiation of INTR_A io-buffer (must be 1 to include io-buffer)	C_INCLUDE_INTR_A_BUF	0 = not included 1 = included	1	integer
G38	Include explicit instantiation of REQ_N io-buffer (must be 1 to include io-buffer)	C_INCLUDE_REQ_N_BUF	0 = not included 1 = included	1	integer
G39	This parameter is no longer active. However, if it is set by the user, it must still have an allowable value.	C_TRIG_IPIF_WRBURST_OCC_LEVEL	2 to the lesser of 24 or the PCI2IPIF FIFO DEPTH-3. PCI2IPIF FIFO DEPTH given by 2^C_PCI2IPIF_FIFO_ABUS_WIDTH	8	integer
G40	IPIF2PCI FIFO occupancy level that starts data transfer (Both as initiator and target on PCI) to PCI agent with multiple data phases per transfer (must meet 16 PCI period maximum).	C_TRIG_PCI_DATA_XFER_OCC_LEVEL	2 to the lesser of 24 or the IPIF2PCI FIFO DEPTH-3. IPIF2PCI FIFO DEPH given by 2^C_IPIF2PCI_FIFO_ABUS_WIDTH	8	integer
G41	Number of PCI retry attempts in IPIF posted-write operations	C_NUM_PCI_RETRIES_IN_WRITES	Any integer	3	integer
G42	Number of PCI clock periods between retries in posted- write operations	C_NUM_PCI_PRDS_BETWN_RETRIES_IN_WRITES	Any integer	6	integer
G43	Device base address	C_BASEADDR	Valid PLB address (1), (2)	0xFFFFFFFF	std_logic_vector
G44	Device absolute high address	C_HIGHADDR	Valid PLB address (1), (2)	0x00000000	std_logic_vector
G45	Include the registers for high-order bits to be substituted in translation	C_INCLUDE_BAROFFSET_REG	1 = include 0 = exclude	0	integer
G46	Include the register for local bridge device number when configuration functionality (C_INCLUDE_PCI_CONFIG =1) is included	C_INCLUDE_DEVNUM_REG	1 = include 0 = exclude	0	integer
G47	Length of PCI Initiated burst reads (in words)	C_PCI_INIT_RD_BURST_LENGTH	2-128	16	integer

Table 1: **PLBV46 PCI Bridge Interface Design Parameters *(Cont'd)***

Generic	Feature / Description	Parameter Name	Allowable Values	Default Value	VHDL Type
G48	PCI initiated prefetch Discard Timer value (power of 2 in PCI clocks)	C_PCI_DISCARD_TIMER	10 or 15	10	integer
G49	PLB initiated prefetch Discard Timer value (power of 2 in SPLB clocks)	C_PLB_DISCARD_TIMER	10 or 15	10	integer
G50	Number of IDELAY controllers instantiated. Ignored if not Virtex-4 or Virtex-5 FPGAs	C_NUM_IDELAYCTRL	2-6 (Virtex-4 or Virtex-5 FPGAs only)	2	integer
G51	Includes IDELAY primitive on GNT_N. Set by TCL-scripts and ignored if not Virtex-4 or Virtex-5 FPGAs.	C_INCLUDE_GNT_DELAY	1=Include IDELAY primitive (Virtex-4 or Virtex-5 FPGAs only) 0=No IDELAY primitive	0	integer
G52	Provides a means for BSB to pass LOC coordinates for IDELAYCTRLs for a given board to EDK and is optional for user to set LOC constraints. This parameter has no impact on bridge functionality.	C_IDELAY CTRL_LOC	See Device Implementation section subsection Virtex-4 and Virtex-5 FPGA Support for allowed values	NOT_SET	string
PCI32 Core Parameters Group					
G53	PCI Configuration Space Header Device ID	C_DEVICE_ID	16-bit vector	0x0000	std_logic_vector
G54	PCI Configuration Space Header Vendor ID	C_VENDOR_ID	16-bit vector	0x0000	std_logic_vector
G55	PCI Configuration Space Header Class Code	C_CLASS_CODE	24-bit vector	0x000000	std_logic_vector
G56	PCI Configuration Space Header Rev ID	C_REV_ID	8-bit vector	0x00	std_logic_vector
G57	PCI Configuration Space Header Subsystem ID	C_SUB SYSTEM_ID	16-bit vector	0x0000	std_logic_vector
G58	PCI Configuration Space Header Subsystem Vendor ID	C_SUBSYSTEM_VENDOR_ID	16-bit vector	0x0000	std_logic_vector
G59	PCI Configuration Space Header Maximum Latency	C_MAX_LAT	8-bit vector	0x0F	std_logic_vector
G60	PCI Configuration Space Header Minimum Grant	C_MIN_GNT	8-bit vector	0x04	std_logic_vector

Table 1: **PLBV46 PCI Bridge Interface Design Parameters *(Cont'd)***

Generic	Feature / Description	Parameter Name	Allowable Values	Default Value	VHDL Type
Configuration					
G61	Include configuration functionality via IPIF transactions	C_INCLUDE_ PCI_CONFIG	0 = Not included 1 = Included	1	integer
G62	Number of IDSEL signals supported	C_NUM_IDSEL	1 to 16	8	integer
G63	PCI address bit that PCI32 core IDSEL is connected to	C_BRIDGE_ IDSEL_ADDR_BIT	31 down to 16 Must be <= 15 + C_NUM_IDSEL. AD(31 down to 0) index labeling	16	integer
IPIF Parameters Group					
G64	PLB master ID bus width (set automatically by XPS)	C_SPLB_MID_ WIDTH	$\log_2$(C_SPLB_NUM_ MASTERS)	3	integer
G65	Number of masters on PLB bus (set automatically by XPS)	C_SPLB_NUM_ MASTERS	1-16	8	integer
G66	PLB Slave Address width	C_SPLB_ AWIDTH	32 (only allowed value)	32	integer
G67	PLB Slave Data width	C_SPLB_ DWIDTH	32 - 128	32	integer
G68	PLB Master Address width	C_MPLB_ AWIDTH	32 (only allowed value)	32	integer
G69	PLB Master Data width	C_MPLB_ DWIDTH	32 - 128	32	integer
G70	The dwidth of the smallest master that accesses the slave IPIF	C_SPLB_ SMALLEST_ MASTER	32 - 128	32	integer
G71	The dwidth of the smallest slave that accesses the master IPIF	C_MPLB_ SMALLEST_ SLAVE	32 - 128	32	integer
G72	Specifies the target technology	C_FAMILY	See PLBV46 IPIF data sheet	virtex4	string

Notes:

1. 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.
2. The minimum address range specified by C_BASEADDR and C_HIGHADDR must be at least 0x1FF. C_BASEADDR must be a multiple of the range, where the range is C_HIGHADDR - C_BASEADDR + 1.
3. The maximum burst size on either the PCI or PLB is limited to the usable FIFO depth which is the physical depth -3.
4. The minimum address range specified by C_IPIFBAR_X and C_IPIF_HIGHADDR_X must be at least 0x1F. C_IPIFBAR_X must be a multiple of the range, where the range is C_IPIF_HIGHADDR_X - C_IPIFBAR_X + 1.

PLBV46 PCI Bus Interface I/O Signals

The I/O signals for the PLBV46 PCI Bridge are listed in Table 2. The interfaces referenced in this table are shown in Figure 1 in the PLBV46 PCI Bridge block diagram.

Table 2: **PLBV46 PCI Bridge I/O Signals**

Port	Signal Name	Interface	I/O	Description
System Signals				
P1	IP2INTC_Irpt	Internal	O	Interrupt from IP to the Interrupt Controller
PLB Signals (Slave)				
P2	SPLB_Clk	PLB Bus	I	PLB slave bus clock. See table note 1.
P3	SPLB_Rst	PLB Bus	I	PLB slave bus reset. See table note 1.
P4	PLB_Abort	PLB Bus	I	Note 1 applies from P4 to P43.
P5	PLB_PAValid	PLB Bus	I	↓
P6	PLB_SAValid	PLB Bus	I	
P7	PLB_ABus(0:C_SPLB_AWIDTH-1)	PLB Bus	I	
P8	PLB_UABus(0:C_SPLB_AWIDTH-1)	PLB Bus	I	
P9	PLB_RNW	PLB Bus	I	
P10	PLB_BE(0:[C_SPLB_DWIDTH/8]-1)	PLB Bus	I	
P11	PLB_Type(0:2)	PLB Bus	I	
P12	PLB_Size(0:3)	PLB Bus	I	
P13	PLB_MasterID(0:C_SPLB_MID_WIDTH-1)	PLB Bus	I	
P14	PLB_MSize(0:1)	PLB Bus	I	
P15	PLB_BusLock	PLB Bus	I	
P16	PLB_LockErr	PLB Bus		
P17	PLB_TAttribute(0:15)	PLB Bus		
P18	PLB_RdBurst	PLB Bus	I	
P19	PLB_WrBurst	PLB Bus	I	
P20	PLB_WrDBus(0:C_SPLB_DWIDTH-1)	PLB Bus	I	
P21	PLB_RdPrim	PLB Bus	I	
P22	PLB_WrPrim	PLB Bus	I	
P23	PLB_RdPendPri(0:1)	PLB Bus	I	
P24	PLB_WrPendPri(0:1)	PLB Bus	I	
P25	PLB_RdPendReq	PLB Bus	I	
P26	PLB_WrPendReq	PLB Bus	I	
P27	PLB_ReqPri(0:1)	PLB Bus	I	
P28	Sl_AddAck	PLB Bus	O	
P29	Sl_Wait	PLB Bus	O	
P30	Sl_Rearbitrate	PLB Bus	O	

Table 2: **PLBV46 PCI Bridge I/O Signals *(Cont'd)***

Port	Signal Name	Interface	I/O	Description
P31	SI_SSize(0:1)	PLB Bus	O	
P32	SI_WrDAck	PLB Bus	O	
P33	SI_WrComp	PLB Bus	O	
P34	SI_WrBTerm	PLB Bus	O	
P35	SI_RdDBus(0:C_SPLB_DWIDTH-1)	PLB Bus	O	
P36	SI_RdDAck	PLB Bus	O	
P37	SI_RdComp	PLB Bus	O	
P38	SI_RdBTerm	PLB Bus	O	
P39	SI_rdWdAddr(0:3)	PLB Bus	O	
P40	SI_MBusy(0:C_SPLB_NUM_MASTERS-1)	PLB Bus	O	
P41	SI_MRdErr(0:C_SPLB_NUM_MASTERS-1)	PLB Bus	O	
P42	SI_MWrErr(0:C_SPLB_NUM_MASTERS-1)	PLB Bus	O	
P43	SI_MIRQ(0:C_SPLB_NUM_MASTERS-1)	PLB Bus	O	Table note 1 applies from P43 to P4.
PLB Signals (Master)				
P44	MPLB_Clk	PLB Bus	I	PLB master bus clock. See table note 1.
P45	MPLB_Rst	PLB Bus	I	PLB master bus reset. See table note 1.
P46	PLB_MAddrAck	PLB Bus	I	Table note 1 applies from P46 to P75.
P47	PLB_MSSize(0:1)	PLB Bus	I	
P48	PLB_MRearbitrate	PLB Bus	I	
P49	PLB_MTimeout	PLB Bus	I	
P50	PLB_MWrDAck	PLB Bus	I	
P51	PLB_MWrBTerm	PLB Bus	I	
P52	PLB_MRdDBus(0:C_MPLB_DWIDTH-1)	PLB Bus	I	
P53	PLB_MRdWdAddr(0:3)	PLB Bus	I	
P54	PLB_MRdDAck	PLB Bus	I	
P55	PLB_MRdBTerm	PLB Bus	I	
P56	PLB_MBusy	PLB Bus	I	
P57	PLB_MRdErr	PLB Bus	I	
P58	PLB_MWrErr	PLB Bus	I	
P59	PLB_MIRQ	PLB Bus	I	
P60	M_Request	PLB Bus	O	
P61	M_Abort	PLB Bus	O	
P62	M_Priority	PLB Bus	O	
P63	M_Buslock	PLB Bus	O	

Table 2: **PLBV46 PCI Bridge I/O Signals *(Cont'd)***

Port	Signal Name	Interface	I/O	Description
P64	M_LockErr	PLB Bus	O	
P65	M_TAttribute(0:15)	PLB Bus	O	
P66	M_Type(0:2)	PLB Bus	O	
P67	M_BE(0:[C_MPLB_ DWIDTH/8]-1)	PLB Bus	O	
P68	M_RNW	PLB Bus	O	
P69	M_UABus(0:C_MPLB_ AWIDTH-1)	PLB Bus	O	
P70	M_ABus(0:C_MPLB_ AWIDTH-1)	PLB Bus	O	
P71	M_MSize(0:1)	PLB Bus	O	
P72	M_size(0:3)	PLB Bus	O	
P73	M_RdBurst	PLB Bus	O	
P74	M_WrBurst	PLB Bus	O	
P75	M_WrDBus(0:C_MPLB_ DWIDTH-1)	PLB Bus	O	Table note 1 applies from P75 to P46.
PCI Address and Datapath Signals				
P76	AD[C_PCI_DBUS_ WIDTH-1:0]	PCI Bus	I/O	Time-multiplexed address and data bus
P77	CBE[(C_PCI_DBUS_ WIDTH/8)-1:0]	PCI Bus	I/O	Time-multiplexed bus command and byte enable bus
P78	PAR	PCI Bus	I/O	Generates and checks even parity across AD and CBE
PCI Transaction Control Signals				
P79	FRAME_N	PCI Bus	I/O	Driven by an initiator to indicate a bus transaction
P80	DEVSEL_N	PCI Bus	I/O	Indicates that a target has decoded the address presented during the address phase and is claiming the transaction
P81	TRDY_N	PCI Bus	I/O	Indicates that the target is ready to complete the current data phase
P82	IRDY_N	PCI Bus	I/O	Indicates that the initiator is ready to complete the current data phase
P83	STOP_N	PCI Bus	I/O	Indicates that the target has requested to stop the current transaction
P84	IDSEL	PCI Bus	I	Indicates that the interface is the target of a configuration cycle
PCI Interrupt Signals				
P85	INTR_A	PCI Bus	O	Indicates that PCI32 interface requests an interrupt
PCI Error Signals				
P86	PERR_N	PCI Bus	I/O	Indicates that a parity error was detected while the PCI32 interface was the target of a write transfer or the initiator of a read transfer
P87	SERR_N	PCI Bus	I/O	Indicates that a parity error was detected during an address cycle, except during special cycles

Table 2: **PLBV46 PCI Bridge I/O Signals *(Cont'd)***

Port	Signal Name	Interface	I/O	Description
PCI Arbitration Signals				
P88	REQ_N	PCI Bus	O	Indicates to the arbiter that the PCI32 initiator requests access to the bus
P89	GNT_N	PCI Bus	I	Indicates that the arbiter has granted the bus to the PCI32 initiator
PCI System Signals				
P90	RST_N	PCI Bus	I	PCI bus reset signal is used to bring PCI-specific registers, sequences, and signals to a consistent state
P91	PCLK	PCI Bus	I	PCI bus clock signal
PCI Bus Internal Arbiter Signals				
P92	INTR_A_int	Internal	O	INT_A available to internal arbiter
P93	REQ_N_toArb	Internal	O	Input from PCI Bus REQ_N available at top-level as output from bridge
P94	FRAME_I	Internal	O	Input from PCI Bus FRAME_N availalble at top-level as output from bridge
P95	IRDY_I	Internal	O	Input from PCI Bus IRDY_N availalble at top-level as output from bridge
User Asserted PCI Interrupt Signal				
P96	Bus2PCI_INTR	Internal	I	Active high signal to asynchronously assert INTR_A. Inverted signal drives INTR_N user application input of PCI core. See PCI core documents for details on INTR_N functionality.
Virtex-4 or Virtex-5 FPGA Only, IDELAY Clock				
P97	RCLK	Internal	I	200 MHz clock input to IDELAY elements of Virtex-4 and Virtex-5 FPGA buffers. Ignored if not Virtex-4 or Virtex-5 architectures.
PCI Bus Monitoring Debug Vector Signal				
P98	PCI_monitor(0:47)	Internal	O	Output vector to monitor PCI Bus.

Note:

1. This function and timing of this signal are defined in the IBM 128-Bit Processor Local Bus Architecture Specification Version 4.6.

The REQ_N_toArb facilitates an interface to an internal (in the FPGA) pci arbiter. The PCI input buffer for GNT_N is removed. This allows an internal connection to GNT_N when using an internal arbiter. When an external arbiter is used, GNT_N_fromArb is not needed.

REQ_N is a 3-stated I/O. The REQ_N_toArb port is available to maintain a PCI core-like interface. The REQ_N_toArb port allows the use of the same port list for PCI bus interface and the UCF for the PCI32 core is the standard file.

The PCI32 core requires that GNT_N be asserted for two clock cycles to initiate a transaction upon receiving grants.

Bus2PCI_INTR is an active High signal. It allows asynchronous assertion of INTR_A on the PCI bus. The signal is driven by user supplied circuitry, such as a PLB GPIO IP core. If it is not connected in the mhs-file, then EDK tools will tie the signal Low. The signal is inverted in the PLBV46 PCI Bridge and AND'd with the bridge interrupt signal (active Low) to drive the INTR_N input of the PCI32 core. This signal then asynchronously drives INTR_A on the PCI bus. See the PCI32 core specifications on INTR_A behavior relative to PCI input INTR_N.

The PCI32 core command register *interrupt disable* bit controls the INTR_A operation andPCI32 core status register Interrupt status bit flags if PCI32 core INTR_A is asserted.

Port and Parameter Dependencies

The dependencies between the IPI v3.0 Bridge design ports, such as the I/O signals, and the parameters are shown in Table 1.

Table 3: **PLBV46 PCI Bridge Parameters-Port Dependencies**

Generic	Parameter	Affects	Depends	Description
Bridge Features Parameter Group				
G1	C_IPIFBAR_NUM	G2-G25		The set of PLB/IPIF BAR-parameters of N = 0 to C_IPIFBAR_NUM-1 are meaningful. When C_IPIFBAR_NUM < 6, the parameters of N = C_IPIFBAR_NUM up to 5 have no effect. If C_IPIFBAR_NUM = 6, the set of PLB/IPIF BAR-parameters of N = 0 to 5 are all meaningful (G2-G25 are meaningful).
G2	C_IPIFBAR_0	G3	G3	G2 to G3 define range in PLB-memory space that is responded to by this device (IPIF BAR)
G3	C_IPIFBAR_HIGHADDR_0	G2	G2	G2 to G3 define range in PLB-memory space that is responded to by this device (IPIF BAR)
G4	C_IPIFBAR2PCIBAR_0		G2, G3 and G45	Meaningful only if G45 = 0 and in this case only high-order bits that are the same in G2 and G3 are meaningful.
G5	C_IPIF_SPACETYPE_0			
G6	C_IPIFBAR_1	G7	G1 and G7	Meaningful only if G1>1, then G6 to G7 define the range in PLB-memory space that is responded to by this device (IPIF BAR)
G7	C_IPIFBAR_HIGHADDR_1	G6	G1 and G6	Meaningful only if G1>1, then G6 to G7 define the range in PLB-memory space that is responded to by this device (IPIF BAR)
G8	C_IPIFBAR2PCIBAR_1		G1, G6, G7 and G45	Meaningful only if G45 = 0 and G1>1. In this case only high-order bits that are the same in G6 and G7 are meaningful.
G9	C_IPIF_SPACETYPE_1		G1	Meaningful only if G1>1
G10	C_IPIFBAR_2	G11	G1 and G11	Meaningful only if G1>2, then G10 to G11 define the range in PLB-memory space that is responded to by this device (IPIF BAR)
G11	C_IPIFBAR_HIGHADDR_2	G10	G1 and G10	Meaningful only if G1>2, then G10 to G11 define the range in PLB-memory space that is responded to by this device (IPIF BAR)
G12	C_IPIFBAR2PCIBAR_2		G1, G10, G11 and G45	Meaningful only if G45 = 0 and G1>2. In this case only high-order bits that are the same in G10 and G11 are meaningful.
G13	C_IPIF_SPACETYPE_2		G1	Meaningful only if G1>2
G14	C_IPIFBAR_3	G15	G1 and G15	Meaningful only if G1>3, then G14 to G15 define the range in PLB-memory space that is responded to by this device (IPIF BAR)
G15	C_IPIFBAR_HIGHADDR_3	G14	G1 and G14	Meaningful only if G1>3, then G14 to G15 define the range in PLB-memory space that is responded to by this device (IPIF BAR)

Table 3: **PLBV46 PCI Bridge Parameters-Port Dependencies *(Cont'd)***

Generic	Parameter	Affects	Depends	Description
G16	C_IPIFBAR2PCIBAR_3		G1, G14, G15 and G45	Meaningful only if G45 = 0 and G1>3. In this case only high-order bits that are the same in G14 and G15 are meaningful.
G17	C_IPIF_SPACETYPE_3		G1	Meaningful only if G1>3
G18	C_IPIFBAR_4	G19	G1 and G19	Meaningful only if G1>4, then G18 to G19 define the range in PLB-memory space that is responded to by this device (IPIF BAR)
G19	C_IPIFBAR_HIGHADDR_4	G18	G1 and G18	Meaningful only if G1>4, then G18 to G19 define the range in PLB-memory space that is responded to by this device (IPIF BAR)
G20	C_IPIFBAR2PCIBAR_4		G1, G18, G19 and G45	Meaningful only if G45 = 0 and G1>4. In this case only high-order bits that are the same in G18 and G19 are meaningful.
G21	C_IPIF_SPACETYPE_4		G1	Meaningful only if G1>4
G22	C_IPIFBAR_5	G23	G1 and G23	Meaningful only if G1=6, then G22 to G23 define the range in PLB-memory space that is responded to by this device (IPIF BAR)
G23	C_IPIFBAR_HIGHADDR_5	G22	G1 and G22	Meaningful only if G1=6, then G22 to G23 define the range in PLB-memory space that is responded to by this device (IPIF BAR)
G24	C_IPIFBAR2PCIBAR_5		G1, G22, G23 and G45	Meaningful only if G45 = 0 and G1=6. In this case only high-order bits that are the same in G22 and G23 are meaningful.
G25	C_IPIF_SPACETYPE_5		G1	Meaningful only if G1=6
G26	C_PCIBAR_NUM	G27-G32		The set of PCI BAR-parameters of N = 0 to C_PCIBAR_NUM-1 are meaningful. When C_PCIBAR_NUM < 3, the parameters of N = C_PCIBAR_NUM up to 2 have no effect. If C_PCIBAR_NUM = 3, the set of PCI BAR-parameters of N = 0 to 2 are all meaningful (G27-G32 are meaningful)
G27	C_PCIBAR2IPIFBAR_0		G28	Only the high-order bits above the length defined by G28 are meaningful
G28	C_PCIBAR_LEN_0			
G29	C_PCIBAR2IPIFBAR_1		G30	Only the high-order bits above the length defined by G30 are meaningful. Not meaningful if G26=1
G30	C_PCIBAR_LEN_1			Not meaningful if G26=1
G31	C_PCIBAR2IPIFBAR_2		G32	Only the high-order bits above the length defined by G30 are meaningful. Not meaningful if G26=1-2
G32	C_PCIBAR_LEN_2			Not meaningful if G26=1-2
G33	C_PCI_ABUS_WIDTH			Only 1 setting
G34	C_PCI_DBUS_WIDTH			Only 1 setting
G35	C_PCI2IPIF_FIFO_ABUS_WIDTH			
G36	C_IPIF2PCI_FIFO_ABUS_WIDTH			

Table 3: **PLBV46 PCI Bridge Parameters-Port Dependencies *(Cont'd)***

Generic	Parameter	Affects	Depends	Description
G37	C_INCLUDE_INTR_A_BUF	P63		If G37 = 0, an io-buffer for P63 is not explicitly instantiated
G38	C_INCLUDE_REQ_N_BUF	P66		If G38 = 0, an io-buffer for P66 is not explicitly instantiated
G39	C_TRIG_IPIF_WRBURST_OCC_LEVEL		G35	Must be set to 2 to the lesser of 24 or the PCI2IPIF FIFO DEPTH-1 where the PCI2IPIF FIFO-1 depth is given by 2^G35
G40	C_TRIG_PCI_DATA_XFER_OCC_LEVEL		G36	Must be set to 2 to the lesser of 24 or the IPIF2PCI FIFO DEPTH-3 where IPIF2PCI FIFO DEPTH given by 2^G36
G41	C_NUM_PCI_RETRIES_IN_WRITES			
G42	C_NUM_PCI_PRDS_BETWN_RETRIES_IN_WRITES			
G43	C_BASEADDR	G44	G44	G43 to G44 define range in PLB-memory space that is responded to by PLBV46 PCI Bridge register address space
G44	C_HIGHADDR	G43	G43	G43 to G44 define range in PLB-memory space that is responded to by PLBV46 PCI Bridge register address space
G45	C_INCLUDE_BAROFFSET_REG	G4, G8, G12, G16, G20 and G24	G1	If G45=1, G4, G8, G12, G16, G20 and G24 have no meaning. The number of registers included is set by G1
G46	C_INCLUDE_DEVNUM_REG	G63	G61, G62	If G61=0, G46 has no meaning. If G46 and G61=1, G63 has no meaning. Meaningful bits in the Device Number register are defined by G62
G47	C_PCI_INIT_RD_BURST_LENGTH			
G48	C_PCI_DISCARD_TIMER			
G49	C_PLB_DISCARD_TIMER			
G50	C_NUM_IDELAYCTRL		G72	If G72 ≠ Virtex-4 or Virtex-5 FPGA, G50 has no meaning
G51	C_INCLUDE_GNT_DELAY		G72	If G72 ≠ Virtex-4 or Virtex-5 FPGA, G51 has no meaning
G52	C_IDELAYCTRL_LOC		G50 and G72	If G72 ≠ Virtex-4 or Virtex-5 FPGA, G52 has no meaning. If G72=Virtex-4 or Virtex-5 FPGA, G52 must include the number of LOC coordinates specified by G50
PCI32 Core Parameters Group				
G53	C_DEVICE_ID			
G54	C_VENDOR_ID			
G55	C_CLASS_CODE			
G56	C_REV_ID			
G57	C_SUBSYSTEM_ID			
G58	C_SUBSYSTEM_VENDOR_ID			
G59	C_MAX_LAT			

Table 3: **PLBV46 PCI Bridge Parameters-Port Dependencies *(Cont'd)***

Generic	Parameter	Affects	Depends	Description
G60	C_MIN_GNT			
Configuration				
G61	C_INCLUDE_PCI_ CONFIG	G62, G63, P84		If G61=1, signal P84 has an internal connection and the top-level port P84 has no internal connection
G62	C_NUM_IDSEL	G49 and G63	G61 and G63	If G61=0, G62 has no meaning. If G61=1, G62 sets the number of devices supported in configuration operations. Must be sufficiently large to include the address bit defined by G63. If G46=1, G62 restricts the allowed values that are meaningful in the Device Number Register
G63	C_BRIDGE_IDSEL_ADDR_BIT	G62	G46, G61 and G62	If G61=0 or G46=1, G63 has no meaning. If G61=1 and G46=0, G63 must be consistent with the setting of G62
IPIF Parameters Group				
G64	C_SPLB_MID_WIDTH			
G65	C_SPLB_NUM_MASTERS			
G66	C_SPLB_AWIDTH			
G67	C_SPLB_DWIDTH			
G68	C_MPLB_AWIDTH			
G69	C_MPLB_DWIDTH			
G70	C_SPLB_SMALLEST_ MASTER			
G71	C_MPLB_SMALLEST_SLAVE			
G72	C_FAMILY	G50-52		If G72 ≠ Virtex-4 or Virtex-5 FPGA, G50-52 have no meanings.

Supported PCI Bus Commands

The list of commands supported by the PCI32 interface is provided in Table 4.

Table 4: **Supported PCI Bus Commands**

Command		PLBV46 PCI Bridge	
Code	**Name**	**Target**	**Initiator**
0000	Interrupt Acknowledge	No	No
0001	Special Cycle	No	No
0010	I/O Read	No	Yes
0011	I/O Write	No	Yes
0100	Reserved	Ignore	Ignore
0101	Reserved	Ignore	Ignore
0110	Memory Read	Yes	Yes
0111	Memory Write	Yes	Yes
1000	Reserved	Ignore	Ignore
1001	Reserved	Ignore	Ignore
1010	Configuration Read	Yes	Optional
1011	Configuration Write	Yes	Optional
1100	Memory Read Multiple	Yes	Yes
1101	Dual Address Cycle	Ignore	No
1110	Memory Read Line	Yes	No
1111	Memory Write Invalidate	Yes	No

PLBV46 PCI Bridge Register Descriptions

The PLBV46 PCI Bridge contains addressable registers for read/write operations as shown in Table 5. The base address for these registers is set by the base address parameter (C_BASEADDR). The address of each register is then calculated by an offset to the base address as shown in Table 5. Registers that reside in the user area of the PCI configuration header are mirrored in the IPIF register space as read-only registers; this is included for debug utility. The registers that exist in a given PLBV46 PCI Bridge depend on the configuration of the bridge.

Table 5: **PLBV46 PCI Bus Interface Registers**

Register Name	PLB Address	Access
Device Interrupt Status Register (ISR)	C_BASEADDR + 0x00	Read/TOW
Device Interrupt Pending Register (IPR)	C_BASEADDR + 0x04	Read
Device Interrupt Enable Register (IER)	C_BASEADDR + 0x08	Read/Write
Device Interrupt ID (IID)	C_BASEADDR + 0x18	Read
Global Interrupt Enable Register (GIE)	C_BASEADDR + 0x1C	Read/Write
Bridge Interrupt Register	C_BASEADDR + 0x20	Read/TOW
Bridge Interrupt Enable Register	C_BASEADDR + 0x28	Read/Write
Reset Module	C_BASEADDR + 0x80	Read/Write
Configuration Address Port	C_BASEADDR + 0x10C	Read/Write
Configuration Data Port	C_BASEADDR + 0x110	Read/Write
Bus Number/Subordinate Bus Number	C_BASEADDR + 0x114	Read/Write
IPIFBAR2PCIBAR_0 high-order bits	C_BASEADDR + 0x180	Read/Write
IPIFBAR2PCIBAR_1 high-order bits	C_BASEADDR + 0x184	Read/Write
IPIFBAR2PCIBAR_2 high-order bits	C_BASEADDR + 0x188	Read/Write
IPIFBAR2PCIBAR_3 high-order bits	C_BASEADDR + 0x18C	Read/Write
IPIFBAR2PCIBAR_4 high-order bits	C_BASEADDR + 0x190	Read/Write
IPIFBAR2PCIBAR_5 high-order bits	C_BASEADDR + 0x194	Read/Write
Host Bridge device number	C_BASEADDR + 0x198	Read/Write

Register and Parameter Dependencies

The addressable registers in the PLBV46 PCI Bridge depend on the parameter settings shown in Table 6.

Table 6: **Register and Parameter Dependencies**

Register Name	Parameter Dependence
Device Interrupt Status Register (ISR)	Always present
Device Interrupt Pending Register (IPR)	Always present
Device Interrupt Enable Register (IER)	Always present
Device Interrupt ID (IID)	Always present
Global Interrupt Enable Register (GIE)	Always present
Bridge Interrupt Register	Always present
Bridge Interrupt Enable Register	Always present
Reset Module	Always present
Configuration Address Port	Present only if G61=1
Configuration Data Port	Present only if G61=1
Bus Number/Subordinate Bus Number	Present only if G61=1
IPIFBAR2PCIBAR_0 High-Order Bits	Present only if G45=1
IPIFBAR2PCIBAR_1 High-Order Bits	Present only if G1>1 and G45=1
IPIFBAR2PCIBAR_2 High-Order Bits	Present only if G1>2 and G45=1
IPIFBAR2PCIBAR_3 High-Order Bits	Present only if G1>3 and G45=1
IPIFBAR2PCIBAR_4 High-Order Bits	Present only if G1>4 and G45=1
IPIFBAR2PCIBAR_5 High-Order Bits	Present only if G1=6 and G45=1
Host Bridge Device Number	Present only if G46=1

PLBV46 PCI Bridge Interrupt Registers Descriptions

The interrupt module registers are always included in the bridge.

Interrupt Module Specifications

The interrupt registers are in the interrupt module that is instantiated in the IPIF module of the PLBV46 PCI Bridge.

Device Interrupt Status Register (DISR)

The Device Interrupt Status Register gives the interrupt status for the device (IPIF + Bridge Interrupts). Each bit within this register represents a major function within the device. The bits are detailed in Table 7. This register is fixed at 32 bits wide and each utilized bit within the register is set to '1' whenever the corresponding interrupt input has met the interrupt capture criteria. Unlike the Bridge Interrupt Status Register, the interrupt capture mode for this register is fixed. The **DPTO** and **TERR** bits are captured with a 'sample and hold high' mode. This means that if the input interrupt is sampled to be '1' at a rising edge of a PLB Clock pulse, the register bit is set to a '1' and 'held' until the User Interrupt Service Routine clears it to a '0'. The remaining bits within the register (**IPIR**) are pass through. When asserted, they are 'held' by the source of the interrupt (Bridge ISR) and therefore an additional sample and hold operation is not necessary in this register. These interrupts must be cleared at the source function.

Table 7: **Device Interrupt Status Register (DISR) Bit Definitions (Bit assignment assumes 32-bit bus)**

Bit(s)	Name	Access	Reset Value	Description
0-28	I	Read	0x0	Unassigned
29	BIR	Read/TOW	0x0	**Bridge Interrupt Request**. This interrupt indicates that the Bridge interrupt input on the IPIF input has been captured in the Bridge Interrupt Status Register and is enabled via the Bridge Interrupt Enable Register. • '0' = No enabled interrupt is captured • '1' = Bridge interrupt is captured and enabled.
30	DPTO	Read/TOW	0x0	**Data Phase Time-out**. This interrupt indicates that a Data Phase time-out occurred during a Read or Write transaction request. The time-out value (PLB clocks) is set to 255 clock periods. • '0' = No Time-out detected. • '1' = Data phase Time-out detected.
31	TERR	Read/TOW	0x0	**Transaction Error.** This interrupt indicates that a function within the Bridge (not IPIF timeout) responded to a Read or Write transaction request with the assertion of the Slv_MErr signal. • '0' = No transaction error detected. • '1' = Transaction Error detected.

Device Interrupt Pending Register (IPR)

The Device Interrupt Pending Register is a read-only value that is the logical AND of the Device Interrupt Status Register and the Device Interrupt Enable Register (see Table 8) on a bit-by-bit basis. The bits are detailed in Table 8. Therefore, the Interrupt Pending Register only reports captured interrupts that are also enabled by the corresponding bit in the Interrupt Enable Register.

Table 8: **Device Interrupt Pending Register (DIPR) Bit Definitions (Bit assignment assumes 32-bit bus)**

Bit(s)	Name	Access	Reset Value	Description
0-28	I	Read	0x0	Unassigned
29	BIRP	Read	0x0	**Bridge Interrupt Pending**. This bit is the logical 'AND' of the Bridge IR bit in the DISR and the corresponding bit in the DIER. • '0' = No Bridge interrupt pending • '1' = Bridge interrupt is pending.
30	DPTOP	Read	0x0	**Data Phase Time-out Pending.** This bit is the logical 'AND' of the DPTO bit in the DISR and the corresponding bit in the DIER • '0' = No Time-out interrupt pending. • '1' = Time-out captured and enabled.
31	TERRP	Read	0x0	**Transaction Error Pending.** This bit is the logical 'AND' of the TERR bit in the DISR and the corresponding bit in the DIER • '0' = No transaction error pending. • '1' = Transaction Error captured and enabled.

Device Interrupt Enable Register (IER)

The Device Interrupt Enable Register determines which interrupt sources in the Device Interrupt Status Register are allowed to generate interrupts to the system. The bits are detailed in Table 9.

Table 9: **Device Interrupt Enable Register (DIER) Bit Definitions (Bit assignment assumes 32-bit bus)**

Bit(s)	Name	Access	Reset Value	Description
0-28	I	Read	0x0	Unassigned (not writable)
29	BIRE	Read/Write	0x0	**Bridge Interrupt Request Enable**. This bit is the interrupt enable for the BIR bit in the DISR. • '0' = Mask Interrupt. • '1' = Enable Interrupt.
30	DPTOE	Read/Write	0x0	**Data Phase Time-out Enable.** This bit is the interrupt enable for the DPTO bit in the DISR. • '0' = Mask Interrupt. • '1' = Enable Interrupt.
31	TERRE	Read/Write	0x0	**Transaction Error Enable.** This bit is the interrupt enable for the TERR bit in the DISR. • '0' = Mask Interrupt. • '1' = Enable Interrupt.

Device Interrupt ID (IID)

The Device Interrupt ID Register is an ordinal value output of a priority encoder. The value indicates which interrupt source, if any, has a pending interrupt. A value of 0x80 indicates that there are no pending interrupts, otherwise, the value gives the bit position in the DIPR of the highest priority interrupt that is pending.

The priority is highest for the interrupt bit in the LSB position (bit 31), which reports as ID value 0x00, and decreases in priority (and increases in the reported ID value) for each successively more significant position, such as going left. The bits are detailed in Table 10.

Table 10: **Device Interrupt ID Register (DIIDR) Bit Definitions (Bit assignment assumes 32-bit bus)**

Bit(s)	Name	Access	Reset Value	Description
0-23		Read	0x0	Unassigned
24-31	IID	Read	0x80	Interrupt ID. • 0x80 - The DIPR has no pending interrupts. • Otherwise - The ordinal ID of the highest-priority pending interrupt in the DIPR.

Global Interrupt Enable Register Description

A global enable is provided to globally enable or disable interrupts from the PCI device. This bit is AND'd with the output to the interrupt controller. Bit assignment is shown in Table 11. Unlike most other registers, this bit is the MSB on the PLB. This bit is read/write and cleared upon reset.

Table 11: **Global Interrupt Enable Register Bit Definitions (Bit assignment assumes 32-bit bus)**

Bit(s)	Name	Access	Reset Value	Description
0	Interrupt Global Enable	Read/Write	0x0	**Interrupt Global Enable-** PLB bit (0) is the Interrupt Global Enable bit. Enables all individually enabled interrupts to be passed to the interrupt controller. • 0 - Not enabled • 1 - Enabled
1-31		Read	0x0	Unassigned-

Bridge Interrupt Register Description

The PLBV46 PCI Bridge has twelve interrupt conditions. The Bridge Interrupt Enable Register enables each interrupt independently. Bit assignment in the Interrupt register for a 32-bit data bus is shown in Table 12. The interrupt register is read-only and bits are cleared by writing a 1 to the bit(s) that are set (1). However, writing a 1 to any bit(s) that are cleared (0) will toggle them to be set (1). All bits are cleared upon reset. For more information, see the PLBV46 IPIF Interrupt Product Specification; the module is labeled PLB Interrupt module, but is used in the PLBV46 IPIF.

Table 12: **Bridge Interrupt Register Bit Definitions (Bit Assignment Assumes 32-bit Bus)**

Bit(s)	Name	Access	Reset Value	Description
0-14		Read	0x0	Unassigned
15	PLB Read Slave BAR Overrun	Read/Write 1 to toggle	0x0	**PLB Read Slave BAR Overrun-** Interrupt(15) indicates the bridge PLB Slave was requested to burst past the BAR limit on a read operation.
16	PLB Write Slave BAR Overrun	Read/Write 1 to toggle	0x0	**PLB Write Slave BAR Overrun-** Interrupt(16) indicates the bridge PLB Slave was requested to burst past the BAR limit on a write operation.
17	PLB Master Read Rearb Timeout	Read/Write 1 to toggle	0x0	**PLB Master Read Rearb Timeout-** Interrupt(17) indicates the bridge PLB Master was rearbitrated 2048 times on a read operation.
18	PLB Master Write Rearb Timeout	Read/Write 1 to toggle	0x0	**PLB Master Write Rearb Timeout-** Interrupt(18) indicates the bridge PLB Master was rearbitrated 2048 times on a write operation.
19	PCI Initiator Write SERR	Read/Write 1 to toggle	0x0	**PCI Initiator Write SERR-** Interrupt(19) indicates a SERR error was detected during a PCI initiator write of data to a PLB slave.
20	PCI Initiator Read SERR	Read/Write 1 to toggle	0x0	**PCI Initiator Read SERR-** Interrupt(20) indicates a SERR error was detected during a PCI initiator read of data from a PLB slave.
21	PLB Master Prefetch Timeout	Read/Write 1 to toggle	0x0	**PLB Master Prefetch Timeout-** Interrupt(21) indicates the PLB Discard timer has timed out, which means the prefetched data was never requested again after the prefetch operation was complete.
22	PLB Master Write Retry Timeout	Read/Write 1 to toggle	0x0	**PLB Master Burst Write Retry Timeout-** Interrupt(22) indicates the automatic PCI write retries were not successful due to a latency timeout on the last retry during a PLB Master burst write to a PCI target.

Table 12: **Bridge Interrupt Register Bit Definitions (Bit Assignment Assumes 32-bit Bus) *(Cont'd)***

Bit(s)	Name	Access	Reset Value	Description
23	PLB Master Write Retry Disconnect	Read/Write 1 to toggle	0x0	**PLB Master Burst Write Retry Disconnect-** Interrupt(23) indicates the automatic PCI write retries were not successful due to a target disconnect on the last retry during a PLB Master burst write to a PCI target.
24	PLB Master Write Retry	Read/Write 1 to toggle	0x0	**PLB Master Write Retry-** Interrupt(24) indicates the automatic PCI write retries were not successful due to a PCI retry on the last retry during a PLB Master burst write to a PCI target.
25	PLB Master Write Master Abort	Read/Write 1 to toggle	0x0	**PLB Master Write Master Abort-** Interrupt(25) indicates that the PLBV46 PCI Bridge asserted a PCI master abort due to no response from a target.
26	PLB Master Write Target Abort	Read/Write 1 to toggle	0x0	**PLB Master Write Target Abort-** Interrupt(26) indicates a PCI target abort occurred during a PLB Master Write to a PCI target.
27	PLB Master Write PERR	Read/Write 1 to toggle	0x0	**PLB Master Write PERR-** Interrupt(27) indicates a PERR error is detected on a PLB Master write to a PCI target.
28	PLB Master Write SERR	Read/Write 1 to toggle	0x0	**PLB Master Write SERR-** Interrupt(28) indicates that a SERR error was detected by the PCI32 core when performing as a PCI initiator writing data to a PCI target.
29	PLB Master Read Target Abort	Read/Write 1 to toggle	0x0	**PLB Master Read Target Abort-** Interrupt(29) indicates that a target abort was detected by the PCI32 core when performing as a PCI initiator reading data from a PCI target.
30	PLB Master Read PERR	Read/Write 1 to toggle	0x0	**PLB Master Read PERR-** Interrupt(30) indicates that a PERR was detected by thePCI32 core when performing as a PCI initiator reading data from a PCI target.
31	PLB Master Read SERR	Read/Write 1 to toggle	0x0	**PLB Master Read SERR-** Interrupt(31) indicates that a SERR error was detected by the PCI32 core when performing as a PCI initiator reading data from a PCI target.

Bridge Interrupt Enable Register Description

The PLBV46 PCI Bridge has interrupt enable features as described in IPSPEC048 PLB Device Interrupt Architecture. Bit assignment in the Bridge Interrupt Enable Register is shown in Table 13. The interrupt enable register is read/write. All bits are cleared upon reset.

Table 13: **Bridge Interrupt Enable Register Bit Definitions (Bit assignment assumes 32-bit bus)**

Bit(s)	Name	Access	Reset Value	Description
0-14		Read	0x0	Unassigned
15	PLB Read Slave BAR Overrun	Read/Write	0x0	**PLB Read Slave BAR Overrun Enable-** Enables this interrupt to be passed to the interrupt controller. • 0 - Not enabled. • 1 - Enabled.
16	PLB Write Slave BAR Overrun	Read/Write	0x0	**PLB Write Slave BAR Overrun Enable-** Enables this interrupt to be passed to the interrupt controller. • 0 - Not enabled. • 1 - Enabled.

Table 13: **Bridge Interrupt Enable Register Bit Definitions (Bit assignment assumes 32-bit bus) *(Cont'd)***

Bit(s)	Name	Access	Reset Value	Description
17	PLB Master Read Rearb Timeout	Read/Write	0x0	**PLB Master Read Rearb Timeout Enable-** Enables this interrupt to be passed to the interrupt controller. • 0 - Not enabled. • 1 - Enabled.
18	PLB Master Write Rearb Timeout	Read/Write	0x0	**PLB Master Write Rearb Timeout Enable-** Enables this interrupt to be passed to the interrupt controller. • 0 - Not enabled. • 1 - Enabled.
19	PCI Initiator Write SERR	Read/Write	0x0	**PCI Initiator Write SERR Enable-** Enables this interrupt to be passed to the interrupt controller. • 0 - Not enabled. • 1 - Enabled.
20	PCI Initiator Read SERR	Read/Write	0x0	**PCI Initiator Read SERR Enable-** Enables this interrupt to be passed to the interrupt controller. • 0 - Not enabled. • 1 - Enabled.
21	PLB Master Prefetch Timeout	Read/Write	0x0	**PLB Master Prefetch Timeout Enable-** Enables this interrupt to be passed to the interrupt controller. • 0 - Not enabled. • 1 - Enabled.
22	PLB Master Write Retry Timeout	Read/Write	0x0	**PLB Master Burst Write Retry Timeout Enable-** Enables this interrupt to be passed to the interrupt controller. • 0 - Not enabled. • 1 - Enabled.
23	PLB Master Write Retry Disconnect	Read/Write	0x0	**PLB Master Burst Write Retry Disconnect Enable-** Enables this interrupt to be passed to the interrupt controller. • 0 - Not enabled. • 1 - Enabled.
24	PLB Master Write Retry	Read/Write	0x0	**PLB Master Write Retry Enable-** Enables this interrupt to be passed to the interrupt controller. • 0 - Not enabled. • 1 - Enabled.
25	PLB Master Write Master Abort	Read/Write	0x0	**PLB Master Write Master Abort Enable-** Enables this interrupt to be passed to the interrupt controller. • 0 - Not enabled. • 1 - Enabled.
26	PLB Master Write Target Abort	Read/Write	0x0	**PLB Master Write Target Abort Enable-** Enables this interrupt to be passed to the interrupt controller. • 0 - Not enabled. • 1 - Enabled.

Table 13: **Bridge Interrupt Enable Register Bit Definitions (Bit assignment assumes 32-bit bus) *(Cont'd)***

Bit(s)	Name	Access	Reset Value	Description
27	PLB Master Write PERR	Read/Write	0x0	**PLB Master Write PERR Enable-** Enables this interrupt to be passed to the interrupt controller. • 0 - Not enabled. • 1 - Enabled.
28	PLB Master Write SERR	Read/Write	0x0	**PLB Master Write SERR Enable-** Enables this interrupt to be passed to the interrupt controller. • 0 - Not enabled. • 1 - Enabled.
29	PLB Master Read Target Abort	Read/Write	0x0	**PLB Master Read Target Abort Enable-** Enables this interrupt to be passed to the interrupt controller. • 0 - Not enabled. • 1 - Enabled.
30	PLB Master Read PERR	Read/Write	0x0	**PLB Master Read PERR Enable-** Enables this interrupt to be passed to the interrupt controller. • 0 - Not enabled. • 1 - Enabled.
31	PLB Master Read SERR	Read/Write	0x0	**PLB Master Read SERR Enable-** Enables this interrupt to be passed to the interrupt controller. • 0 - Not enabled. • 1 - Enabled.

PLBV46 PCI Bridge Reset Register Description

The IP Reset module is always instantiated in the PLBV46 PCI Bridge. Details on the IPIF Reset module can be found in the *Processor IP Reference Guide*. The IP Reset module permits the software reset of the PLBV46 PCI Bridge, independently of other modules in the system. However, the PCI32 core is not reset by the software reset and can only be reset by the PCI bus RST_N signal. The MIR is not included.

Configuration Address Port Register Description

The Configuration Address Port Register exists only if the bridge is configured with PCI host bridge configuration functionality, such as C_INCLUDE_PCI_CONFIG=1. This register is read/write with some bits hardwired as in Table 14. Definition of this register is a subset of the PCI 2.2. All accesses to the register are 32-bit accesses. Data is latched on a write in all 32-bits except where bits are hard-wired. A read yields all 32-bits. Reset clears all bits. Eight and sixteen bit accesses are not supported, therefore, such accesses are not passed on as I/O accesses. Byte address integrity is maintained from PCI little endian word format when writing/reading data to/from the Configuration Address Port Register which is defined in big endian word format.

Table 14: **Configuration Address Port Register Bit Definitions (Bit assignment assumes 32-bit bus)**

Bit(s)	Name	Access	Reset Value	Description
0-5	D0-D5	Read/Write	0x0	Identifies the target word address (32bits) within the function's configuration space (1-64)
6-7	D6-D7	Read	0x0	Hard-wired to 0, read-only
8-12	D8-D12	Read/Write	0x0	Identifies the target PCI Device (0-31)
13-15	D13-D15	Read/Write	0x0	Identifies the target function (1-8)
16-23	D16-D23	Read/Write	0x0	Identifies the target PCI Bus (1-256)
24	D24	Read/Write	0x0	Active high enable bit
25-31	D25-D31	Read	0x0	Reserved and hardwired to 0.

Configuration Data Port Register Description

The Configuration Data Port Register exists only if the bridge is configured with PCI host bridge configuration functionality, such as C_INCLUDE_PCI_CONFIG=1. This register is read/write and definition of this register follows PCI 2.2. All accesses to the register are 32-bit accesses. A read initiates a configuration read command and a write initiates a configuration write command. Determination of whether the command is a type 0 or type 1 depends on the comparison results of the bus number compare. The fields are defined in Table 15. Reset clears all bits. Byte address integrity is maintained from PCI little endian word format when writing/reading data to/from the Configuration Data Port register which is defined in big endian word format.

Table 15: **Configuration Data Port Address Register Bit Definitions (Bit Assignment Assumes 32-bit Bus)**

Bit(s)	Name	Access	Reset Value	Description
0-31	D0 - D31	Read/Write	0x0	Read or write causes automatic execution of Configuration Read Command or Configuration Write Command using address/bus information in the Configuration Address Port register.

Bus Number/Subordinate Bus Number Register Description

The Bus Number/Subordinate Bus Number Register exists only if the bridge is configured with PCI host bridge configuration functionality, such as C_INCLUDE_PCI_CONFIG=1. This register is read/write. All accesses to the register are 32-bit accesses. The bus number is an 8-bit value defining the primary bus number. The highest subordinate bus number is also an 8-bit value. The fields are defined in Table 16. Reset clears all bits.

Table 16: **Bus Number/Subordinate Bus Number Register Bit Definitions (Bit Assignment Assumes 32-bit Bus)**

Bit(s)	Name	Access	Reset Value	Description
0-7	D0- D7	Read	0x0	Reserved
8-15	D8 - D15	Read/Write	0x0	Bus number
16-23	D16 - D23	Read	0x0	Reserved
24-31	D24 - D31	Read/Write	0x0	Maximum subordinate bus number

IPIFBAR2PCIBAR_N High-Order Bits Register Description

When configured to include these registers, such as C_INCLUDE_BAROFFSET_REG=1, the values in the registers are used to translate addresses on the PLB bus to the PCI. The register values are used instead of the corresponding parameter C_IPIFBAR2PCIBAR_N for translation by high-order bit substitution. The parameters C_IPIFBAR2PCIBAR_N have no effect on the bridge operation if the registers for address translation are included.

The number of registers present is given by the number of IPIF BAR configured in the IPIF (C_IPIFBAR_NUM). The actual width of the Nth register is given by the number of high-order bits that define the complete address range corresponding to the Nth IPIF BAR. When the register is read, 32-bits are returned with the low-order bits hard-wired to zero.

The IPIFBAR2PCIBAR_N registers are included in the bridge via the parameter C_INCLUDE_BAROFFSET_REG.

These read/write registers allow dynamic, run-time changes of the high-order bits for the substitution in the translation of an address from the PLB bus to the PCI bus. Low-order bits pass directly from the PLB bus to the PCI bus. When the register is read, 32-bits are read with the low-order bits set to zero. Table 17 shows the data format. The programmability of these registers allows PLB address transactions to access any target on the PCI bus which has been arbitrarily assigned a PCI BAR by a remote or local Host Bridge. Dynamic, run-time changes in the high-order bits for address translation of PLBV46 PCI Bridge PCI BAR range translation to PLB slaves is not needed because the PLB slave addresses are defined at build time.

Including these registers makes the parameters, C_IPIFBAR2PCIBAR_N, irrelevant because the value in the Nth programmable register replaces the values of the corresponding parameter, C_IPIFBAR2PCIBAR_N, in translating the PLB address to the PCI bus. When the registers are included, the parameters, C_IPIFBAR2PCIBAR_N, for N=0 to C_IPIFBAR_NUM-1, have no effect.

Table 17: **IPIFBAR2PCIBAR_N High-Order Bits (Bit assignment assumes 32-bit bus)**

Bit(s)	Name	Access	Reset Value	Description
0-M	D0 - DM	Read/Write	0x0	M+1 high-order bits that are substituted in address translation from Nth IPIFBAR access to PCI address space
M+1-31	DM+1 - D31	Read Only	0x0	Low-order bits set to zero

The following example shows how the IPIFBAR2PCIBAR_N registers assignments define translation of PLB addresses within the range of a given IPIFBAR to PCI address space.

Setting C_INCLUDE_BAROFFSET_REG=1 includes high-order bit registers for all IPIFBARs defined by C_IPIFBAR_NUM.

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

C_IPIFBAR_0=0x12340000
C_IPIF_HIGHADDR_0=0x1234FFFF
C_IPIFBAR2PCIBAR_0=Don't care
C_IPIF_SPACETYPE_0=1

C_IPIFBAR_1=0xABCDE000
C_IPIF_HIGHADDR_1=0xABCDFFFF
C_IPIFBAR2PCIBAR_1=Don't care
C_IPIF_SPACETYPE_1=0

C_IPIFBAR_2=0xFE000000
C_IPIF_HIGHADDR_2=0xFFFFFFFF
C_IPIFBAR2PCIBAR_2=Don't care
C_IPIF_SPACETYPE_2=1

C_IPIFBAR_3=0x00000000
C_IPIF_HIGHADDR_3=0x0000007F
C_IPIFBAR2PCIBAR_3=Don't care
C_IPIF_SPACETYPE_3=1

Associated with each IPIF BAR for C_IPIFBAR_N for N=0 to 3 are four registers for the high-order bits to be substituted when making the translation to PCI memory and /IO space. For the previous example, the following registers are set.

Register for C_IPIFBAR_0 (IPIFBAR2PCIBAR_0 High-Order Bit Register):
Programmable register for 16 high-order bits. The data in the register is substituted for the 16 msb of the address that is translated to PCI bus.

Register for C_IPIFBAR_1 (IPIFBAR2PCIBAR_1 High-Order Bit Register):
Programmable register for 19 high-order bits. The data in the register is substituted for the 19 msb of the address that is translated to PCI bus.

Register for C_IPIFBAR_2 (IPIFBAR2PCIBAR_2 High-Order Bit Register):
Programmable register for 7 high-order bits. The data in the register is substituted for the 7 msb of the address that is translated to PCI bus.

Register for C_IPIFBAR_3 (IPIFBAR2PCIBAR_3 High-Order Bit Register):
Programmable register for 25 high-order bits. The data in the register is substituted for the 25 msb of the address that is translated to PCI bus.

The remaining low-order bits are set to zero when a read of these registers is performed.

Writing 0x56710000 to IPIFBAR2PCIBAR_0 High-Order Bit Register and then accessing the PLBV46 PCI Bridge IPIFBAR_0 with address 0x12340ABC on the PLB bus would yield 0x56710ABC on the PCI bus.

Writing 0xFEDC0000 to IPIFBAR2PCIBAR_1 High-Order Bit Register and then accessing the PLBV46 PCI Bridge IPIFBAR_1 with address 0xABCDF123 on the PLB bus would yield 0xFEDC1123 on the PCI bus.

Writing 0x40000000 to IPIFBAR2PCIBAR_2 High-Order Bit Register and then accessing the PLBV46 PCI Bridge IPIFBAR_2 with address 0xFFFEDCBA on the PLB bus would yield 0x41FEDCBA on the PCI bus.

Writing 0x12345680 to IPIFBAR2PCIBAR_3 High-Order Bit Register and then accessing the PLBV46 PCI Bridge IPIFBAR_3 with address 0x0000004A on the PLB bus would yield 0x123456CA on the PCI bus.

Host Bridge Device Number Register Description

The Host Bridge Device Number register is included by setting C_INCLUDE_DEVNUM_REG=1. The register can be included only if configuration functionality, such as C_INCLUDE_PCI_CONFIG=1, is included.

This register is read/write and is four bits wide. Table 18 shows specifics of the data format. The programmability of this register allows programmable definition of the bridge device number and corresponding address bit that is internally connected to its IDSEL signal. The maximum value that can be loaded in this register is given by the value set by parameter C_NUM_IDSEL minus 1 because the device number must be consistent with the number of devices that are supported in configuration transactions.

Table 18: **Host Bridge Device Number (Bit assignment assumes 32-bit bus)**

Bit(s)	Name	Access	Reset Value	Description
0-27	D0-D27	Read Only	0x0	Set to zero.
28-31	D28 - D31	Read/Write	0x0	Defines the device number of the PLBV46 PCI Bridge when configured as a Host Bridge.

PLB PCI Transactions

The following subsections discuss details of the following types of transactions for the PLBV46 PCI Bridge to realize data throughputs as high as 132 MB/sec. This assumes the PLB clock is 100 MHz or higher. Lower data rates are realized with lower PLB clock rates for some transactions.

- The section, PLB Master Initiates a Read Request of a PCI Target, discusses the PLB master read of a PCI target where the PCI32 core is the PCI initiator.
- The section, PLB Master Initiates a Write Request to a PCI Target, discusses the PLB master write to a PCI target where the PCI32 core is the PCI initiator.
- The section, PCI Initiator Initiates a Read Request of a PLB Slave, discusses the remote PCI initiator read of a PLB device where the PCI32 core is the PCI target
- The section, PCI Initiator Initiates a Write Request to a PLB Slave, discusses the remote PCI initiator write to a PLB device where the PCI32 core is the PCI target.
- The section, Configuration Transactions, discusses PLB master read and write of a PCI target configuration space where the PCI32 core is the PCI initiator.

PLB transactions that are supported are limited to the subset of PLB transactions that are supported by the IPIF. This limitation is caused by the time-multiplexed architecture of the PCI bus where addressing is required to be incremented by 4 bytes per data phase. When operating as a master, the IPIF can either perform single transactions (1-4 bytes) or bursts of an arbitrary length. In the case of writes, the length is determined by the PCI initiator supplying the data and/or by how fast the PCI initiator supplies the data. In the case of reads, the length is determined by either a parameterized number or up to the range limit of the PCI BAR, whichever is less. When the IPIF is operating as a PLB slave, it performs single transfers of 1-4 bytes, burst transfers of any number of words, and 4, 8 or 16-word line transactions. The IPIF always performs line read requests on the IPIC with the address double word aligned, independent of the target word requested.

This is required because the PCI time-multiplexed address and data bus requires sequential addressing. PCI commands that are supported include I/O read, I/O write, memory read, memory write, memory read multiple, memory read line, and memory write invalidate. Table 19 shows the translations of PLB transactions to PCI commands, while in Table 20 shows the translations of PCI commands to PLB transactions.

The PCI transactions that are supported is limited to a subset of all PCI transactions because some features on the PCI are not supported on the PLB. Specifically, dynamic byte enable during multiple data phase transfers is not supported in burst transactions on the PLB. The PLB supports only full words in burst read and write transactions. It is the user's responsibility to ensure that all byte enables are asserted for remote PCI initiator transactions with multiple data phases.

Table 19: **Translation Table for PLB transactions to PCI commands**

Remote PLB Master Transaction	PCI I/O Space Prefetchable or Non-prefetchable	PCI Memory Space Prefetchable	PCI Memory Space Non-prefetchable
Single Read (<=4 bytes)	I/O Read	Memory Read	Not Supported
Read Burst transfer word	I/O Read	Memory Read Multiple	Not Supported
Sequential Read, 4, 8 and 16-word cacheline read (1)	I/O Read	Memory Read Multiple	Not Supported
Single Write (<=4 bytes)	I/O Write	Memory Write	Not Supported
Write Burst transfer word	I/O Write	Memory Write (multiple data phase)	Not Supported
Sequential fill, 4, 8 and 16-word cacheline write (2)	I/O Write	Memory Write (multiple data phase)	Not Supported

Notes:

1. The data is returned sequentially, starting at the first word of the line. This is independent of the target word presented.
2. On write, the 405 always sources the first word, for example, sequential fill, on the line.

Table 20: **Translation Table for PCI commands to PLB transactions**

PCI Initiator Command	PLB Memory Prefetchable	PLB Memory Non-prefetchable
I/O Read	Not Supported	Not Supported
I/O Write	Not Supported	Not Supported
Memory Read (single data phase)	PLB Single Read	Not Supported
Memory Read (multiple data phase)	PLB Burst Read with all BE asserted (1)	Not Supported
Memory Read Multiple	PLB Burst Read with all BE asserted (1)	Not Supported
Memory Read Line	PLB Single Read	Not Supported
Memory Write (single data phase)	PLB Single Write	Not Supported
Memory Write 2 (multiple data phase)	PLB Burst Write of length defined by available data in FIFO (2)	Not Supported
Memory Write Invalidate	PLB Burst Write	Not Supported

Notes:

1. The PLB does not support dynamic byte enable (BE) in burst read transactions so when Memory Read Multiple is translated to a PLB burst read, all BE are asserted during the PLB read operation.
2. The PLB does not support dynamic byte enable (BE) in burst write transactions so when Memory Write Multiple is translated to a PLB burst write, all BE are asserted during the PLB write operation.

For all the transactions listed previously, these design requirements are specified:

- Both PCI and PLB clocks are independent global buffers. For Virtex®-4 or Virtex-5 FPGAs, RCLK must also be driven by a global buffer.
- The PLB clock can be slower or faster than the PCI clock. The ratio of (PLB clock) / (PCI clock) is limited to 100/15 = 6.67, for example, if LB clock = 100 MHz, the PCI clock must be no less than 15 MHz. For Virtex-4 or Virtex-5 FPGAs, RCLK must be 200 MHz.
- Address space on the PCI side accessible from the PLB side must be translated to a 2^N contiguous block on the PLB side. Up to six independent blocks are possible. Each block has parameters for base address (BAR), high address which must define a 2^N range, address translation vector, and memory designator (memory or I/O).
- All address space on the PLB side that is accessible from the PCI side must be translated to a maximum of three 2^N contiguous blocks on the PCI side. Up to three independent blocks are possible because the PCI32 core supports up to 3 BARs. Each block has parameters for length which must be a 2^N range, and address translation vector. Only memory space in the sense of PCI memory space is supported. Space type is mirrored in the PCI configuration registers.
- Address translations in both directions are performed by high-order address bits substitution in the address vector before crossing to the other bus domain. Byte addressing integrity is maintained between buses.
- The user's system must be designed to accommodate certain restrictions on throttling by the PLBV46 PCI Bridge. Both PLB and PCI burst transactions can be broken up into multiple transactions on the target or slave bus due to restrictions on bus protocol and modules in the PLBV46 PCI Bridge. Additional PLB and PCI transactions are automatically initiated when needed to complete a transaction. The first restriction is that the PCI32 core does not permit throttling of data as either the initiator or target except for insertion of wait states prior to the first data transfer. Another restriction is, that as a master on the PLB, the PLBV46 PCI Bridge is not allowed to throttle, but the PCI remote initiator can cause the need to throttle on the PLB. This is particularly true when the PCI clock is significantly slower than the PLB clock. The PLBV46 PCI Bridge circumvents the throttling limitations by terminating transactions as needed and reinitiating the request to continue as needed. Parameters allow the user to optimize the burst size for high data throughput and minimizing the number of transactions needed to complete the desired burst transactions.
- The interrupt status register in the IPIF contains information to identify an error conditions during the implementation of the PLBV46 PCI Bridge and the troubleshooting of the system. To clear the interrupt register bits that were *set* with an error condition, a write of a "1" to the bit position corresponding to the operation must be performed.
- The PCI32 core does not permit throttling of data at either the initiator or target except for insertion of wait states prior to the first data transfer. Consequently, if the PLB device requires throttling that affects the PCI transaction, the PLBV46 PCI Bridge must terminate the transaction. If the PCI32 core is the initiator, a new PCI transaction must be initiated to continue data transfer. Although PLB masters are not allowed to throttle data flow, the combined IPIF and PLBV46 PCI Bridge operation can result in the need for throttling data on the PCI bus, especially when the PLB clock is slower than the PCI clock. The PLBV46 PCI Bridge handles throttling by terminating initiator transactions as needed and continuing the PLB master request with a new PCI transaction. Similarly, new PLB transactions are automatically initiated when needed to complete a PCI initiator transaction.

PLB Master Initiates a Read Request of a PCI Target

This section discusses the operation of a PLB master initiating single, burst and cacheline reads of a remote PCI target. Cacheline reads return the data sequentially, starting at the first word of the line. In these transactions, the PCI32 core is the PCI initiator.

The operation is similar whether the PCI space is memory or I/O space with the exception of the command sent to the PCI32 core. A parameter associated with each BAR must be consistent with the remote PCI device memory type as either I/O or memory. Based on this parameter setting, either I/O or memory commands are asserted. The PLBV46 IPIF and bridge can accept both fixed length and arbitrary length burst transactions on the PLB, such as when burst length is determined by the PLB_rdBurst signal. Only one PLB master read of a PCI target is supported at a time.

Commands supported in PLB master read operations are I/O read, memory read, and memory read multiple. The command used is based on the address and qualifier decode, which includes the address, memory type, such as I/O or memory type, and if burst is asserted. Table 19 shows translations of PLB transactions to PCI commands.

The address presented on the PLB is translated to the PCI address space by high-order bit substitution with the 2 lsbs set as follows:

- If the target PCI address space is memory space, the 2 LSBs are set to 00, as in the linear incrementing mode.
- If the PCI target address space is IO-space, the 2 LSBs are passed unchanged from that presented on the PLB bus.

When the PLBV46 PCI Bridge decodes a PLB read that is for a remote PCI Target, the transfer is rearbitrated on the PLB bus until the requested data has been prefetched from the remote PCI Target.

If the PLB transaction is not a burst, as when PLB_rdBurst is not high) a single PCI transaction (I/O or Memory Read command) is performed. After this transaction is successfully completed, a subsequent PLB single read with the same PLB address and qualifiers then completes on the PLB bus. If the transaction is a PLB burst transaction, as when PLB_rdBurst is high, and the space type is memory, the PLBV46 PCI Bridge issues a memory read multiple command on the PCI bus. After this transaction is successfully completed, a subsequent PLB burst read with the same PLB address and qualifiers then completes on the PLB bus. The number of 32-bit data words read from the remote PCI Target is determined by the encoded length specified in the PLB fixed-length burst transfer. A Discard Timer is used to determine how long the PLBV46 PCI Bridge should wait for a subsequent PLB read with the same PLB address and qualifiers before discarding the prefetched data in the FIFO.

Dynamic byte enable is not supported in Xilinx PLB burst operations and is not supported in the PLB Master read of a PCI target.All byte enable bits are asserted in PLB master burst read operations.

To comply with the PCI specification, PLB masters are required to re-issue commands when a PCI retry is asserted. PCI retries are communicated to the PLB master by asserting PLB rearbitrate without an interrupt.

It is the responsibility of the master to properly read data from non-prefetchable PCI targets. For example, the master must perform single transaction reads of non-prefetchable PCI targets to avoid destructive read operations of a PCI target.

Abnormal Terminations

In the context of the PLBV46 PCI Bridge, cacheline transactions are special cases of a burst. Abnormal terminations during a cacheline read operation have the same response as a burst read transaction.

- If a parity error occurs during the address phase of the prefetch, the PLBV46 PCI Bridge asserts the PLB Master Read SERR interrupt. If the remote PCI target follows the response recommended by the PCI specification to not claim the transactions, the PLBV46 PCI Bridge terminates the transaction with a master abort. If the target does not follow PCI specification recommendation and transfers data, the received data is discarded and not available to the remote PLB Master. Sl_MRdErr is asserted at the first opportunity.
- If a SERR occurs during a valid data phase on a single transfer, the PLBV46 PCI Bridge asserts the PLB Master Read SERR interrupt. The received data is discarded and not available to the remote PLB Master. Sl_MRdErr is asserted at the first opportunity.
- If a SERR occurs during a valid data phase on a burst transfer, the PLBV46 PCI Bridge asserts the IPIF Master Read SERR interrupt. SERR error on data phase could occur on the first PCI transaction or on a subsequent transaction due to an abnormal disconnect that allowed automatic reissue of the PCI read command. The received data is discarded and not available to the remote PLB Master. Sl_MRdErr is asserted at the first opportunity.
- If the PLBV46 PCI Bridge performs a master abort due to no response from a target, the prefetch is abandoned. Sl_MRdErr is asserted at the first opportunity.
- If on either a single transfer or the first data phase of a burst transfer, a PCI retry from the PCI target occurs, the PLBV46 PCI Bridge immediately retries the read request and continue retrying the request until the transfer completes.
- If during a single transfer the target disconnects with data, the transfer will be completed.
- If on a single transfer, a PERR error is detected, data is transferred and the PLB Master Read PERR interrupt is asserted.
- If the target disconnects on a burst transfer, either with or without data, the PCI32 core terminates the PCI transaction. Another PCI transaction is attempted as long as the encoded length specified in the PLB fixed-length burst transfer has not been prefetched.
- If a PERR error is detected on a burst transfer, the PLBV46 PCI Bridge aborts the PCI transaction. Any received data is discarded and not available to the remote PLB Master. The PLB Master Read PERR interrupt is asserted. Sl_MRdErr is asserted at the first opportunity.
- If the initiator latency timer expires on a burst transfer, the PLBV46 PCI Bridge terminates the PCI transaction. Another PCI transaction is attempted as long as the encoded length specified in the PLB fixed-length burst transfer has not been prefetched.
- If a target abort occurs, the PLB Target Abort Master Read interrupt is asserted. Any received data is discarded and not available to the remote PLB Master. Sl_MRdErr is asserted at the first opportunity. Recall that a target abort indicates that the target cannot proceed with subsequent transactions; this is expected to be a major failure most likely requiring a reset.
- If a PLB read request indicates a burst length that extends beyond the valid range of the IPIF BAR, as defined by the C_IPIF_HIGHADDR_X parameter, the PLB Read Slave BAR Overrun interrupt is asserted. The PLBV46 PCI Bridge does not initiate a read on the PCI bus and responds to the PLB Master with Sl_MRdErr asserted with Sl_rdDAck.

Table 21 summarizes the abnormal conditions with which a PCI target can respond and how the response is translated to the PLB master.

Table 21: **Response of PLB Master/v3.0 Initiator read of a remote PCI target with abnormal condition on PCI bus**

Abnormal Condition	Single Transfer	Burst (PLB_rdBurst asserted)
SERR (includes parity error on address phase)	PLB Master Read SERR interrupt asserted. Data is discarded. SI_MRdErr is asserted.	PLB Master Read SERR interrupt asserted. Data is discarded. SI_MRdErr is asserted.
PLBV46 PCI Bridge Master abort (no PCI target response)	Prefetch abandoned. SI_MRdErr is asserted.	Prefetch abandoned. SI_MRdErr is asserted.
Target disconnect without data (PCI Retry)	Immediate automatic retry	Immediate automatic retry
Target disconnect without data (after one completed data phase)	N/A	Data is being buffered in PLBV46 PCI Bridge PCI2IPIF FIFO. The PCI transaction is terminated by the disconnect. If the encoded length has not been prefetched, the PLBV46 PCI Bridge issues another PCI transaction at correct address. If a PCI retry is asserted, the PCI read automatically retried.
Target disconnect with data	Completes	
PERR	Data is transferred and the PLB Master Read PERR interrupt asserted	PLB Master Read PERR interrupt is asserted and any data is discarded
Latency timer expiration	N/A because the PCI32 core waits for one transfer after timeout occurs	Same as target disconnect with/without data
Target Abort	The PLB Target Abort Master Read interrupt asserted. SI_MRdErr is asserted.	The PLB Target Abort Master Read interrupt asserted and any data is discarded. SI_MRdErr is asserted.
Address increments beyond valid range	N/A	Stop PCI transaction. Assert PLB Read Slave BAR Overrun interrupt and assert SI_MRdErr with SI_rdDAck to PLB master.

PLB Master Initiates a Write Request to a PCI Target

This section discusses the operation of an PLB master initiating single, burst and cache line write transactions to a remote PCI target. All PLB write transactions are posted-writes. Because both single PLB writes and burst PLB writes to the bridge are fire-and-forget, any error in completing the write occurs mostly likely after the PLB transaction is completed. The errors are signaled by an interrupt when an incomplete PCI transactions occur or when PCI errors occur. Details of the abnormal terminations are discussed in a later section. In these transactions, the PCI32 core is the PCI initiator.

The operation is essentially the same whether the PCI space is memory or I/O space; the only difference is the command sent to the PCI32 core by the PLBV46 PCI Bridge. The bridge can accept only fixed length burst transactions on the PLB. All PLB burst transfers are 32-bits per data phase; dynamic byte enable is not supported by the PLB protocol. The PLB specification requires all cacheline write transactions to be sequential fill type, independent of the target word; however, the PLBV46 IPIF requires the address received during a cacheline write operation to be the first word of the line being written.

Commands supported in PLB master write operations are I/O write and memory write (both single and burst). The command used is based on the address/qualifier decode, which includes the address, memory type, such as I/O or memory type, and if PLB_wrBurst is asserted. Table 19 shows translations of PLB transactions to PCI commands.

The address presented on the PLB is translated to the PCI address space by high-order bit substitution with the 2 LSBs set as follows. If the target PCI address space is memory space, the 2 LSBs are set to 00, as in the linear incrementing mode. If the PCI target address space is IO-space, the 2 LSBs are passed unchanged from that presented on the PLB bus.

Both single and burst write transfers are posted so the data is buffered in the IPIF2PCI FIFO, which has a depth defined by the parameter C_IPIF2PCI_FIFO_ABUS_WIDTH. Due to the FIFO backup requirement of the PCI32 core, the FIFO usable buffer depth is the actual depth minus 3 words.

Data is loaded in the FIFO on each clock cycle that the write request is asserted and the address decode is valid. If the transaction is not a burst, as when PLB_wrBurst is not high, and the PLB transfer is a single word or bytes within a single word, a single PCI transaction (I/O or Memory Write command) is performed. In PLB burst transfers, as when PLB_wrBurst is asserted, the data is buffered and the PCI transfer is initiated when the PLB write is completed.

Only one PLB master write to a PCI target is supported at a time. Write transactions are not queued in the bridge. After the PLB write to the bridge is completed and while a write to PCI is being completed, the PLBV46 PCI Bridge asserts PLB rearbitrate to terminate subsequent PLB transactions. When a posted write is complete, another write request from a PLB master can be initiated.

Consistent with the PCI specification, the PLBV46 PCI Bridge re-issues commands when an PCI retry is asserted. To avoid permanent livelock, the posted write is attempted to be completed up to a predefined number of retries defined by the parameter C_NUM_PCI_RETRIES_IN_WRITES.

Re-issuing the write operation on the PCI is automatic.

It is the responsibility of the master to properly write data to a PCI target from non-prefetchable PLB sources. For example, it must perform single transaction reads of non-prefetchable PLB sources to avoid loss of data in fire-and-forget writes to a PCI target.

The PLBV46 PCI Bridge does not support fast back-to-back PCI transactions.

Abnormal Terminations

In the context of the PLBV46 PCI Bridge, cacheline transactions are special cases of a burst. Abnormal terminations during a cacheline write operation have the same response as a burst write transaction. Recall that the PLBV46 IPIF specification requires that the targetword of a cacheline write be the first word of the line.

- If a SERR error, including a parity error during the address phase, is detected on either a single or burst transfer, the PLB Master Write SERR interrupt is asserted. If the PLB transfer is in progress, Sl_MWrErr is asserted with Sl_wrDAck.
- If on either a single or burst write the PLBV46 PCI Bridge asserts a master abort due to no response from a target, the PLBV46 PCI Bridge asserts a PLB Master Write Master Abort interrupt. The IPIF2PCI FIFO is flushed when the Master Abort Write interrupt is asserted. If the PLB transfer is in progress, Sl_MWrErr is asserted with Sl_wrDAck.
- If on a single transfer or on the first data cycle of a burst transfer a PCI retry from the PCI target occurs, the PLBV46 PCI Bridge automatically performs up to a parameterized number of retries. The number of retries is set by C_NUM_PCI_RETRIES_IN_WRITES. A parameterized wait time before a retry occurs is set by C_NUM_PCI_PRDS_BETWN_RETRIES_IN_WRITES. Both parameters are set at build time. During the time retries are possible, subsequent PLB master write operations to a PCI target are inhibited by assertion of PLB rearbitrate. If the retries are not successful, as when disconnects or more PCI retries occur, a PLB Master Write interrupt identifying the failure mode is asserted. The IPIF2PCI FIFO is flushed upon asserting any of the three PLB Master Write Retry interrupts. Consistent with the PCI Spec, the PLB master is required to perform the write again if the last of the automatic retries was terminated with a PCI retry.

- If on a single transfer the target disconnects with data, the transfer is completed.
- If the target disconnects, either with or without data after the first data phase of a burst transfer, the IPIF/PCI core terminates the PCI transaction. If the IPIF2PCI FIFO is not empty, another PCI transaction is attempted. Due to pipelining in the PCI core, the IPIF2PCI_FIFO must backup 1-3 words, depending on the type of target disconnect. The PLBV46 PCI Bridge performs up to a parameterized number of retries (C_NUM_PCI_RETRIES_IN_WRITES). A parameterized wait time (C_NUM_PCI_PRDS_BETWN_RETRIES_IN_WRITES) before a retry occurs is included. Both parameters are set at build time and are the same as defined for PCI retry situation. During the time retries are in progress, subsequent PLB master write operations to a PCI target are inhibited. If the PCI transaction retries are not successful due to any combination of PCI retries, disconnection, or time out, a PLB Master Write Retry interrupt, PLB Master Write Retry Disconnect interrupt, or PLB Master Write Retry Timeout interrupt, respectively, is asserted. The actual interrupt that is asserted is defined by the type of disconnect that occurred on the last of the prescribed number of retries. The IPIF2PCI FIFO is flushed upon asserting one of the PLB Master Write interrupts. Consistent with the PCI Spec, the PLB master is required to perform the write again if the last of the automatic retries was terminated with a PCI retry.
- If on a single transfer or on a burst transfer a PERR error during data phase is detected, the PLBV46 PCI Bridge aborts the PCI transaction and a PLB Master Write PERR interrupt is asserted. If the burst transfer is still in progress, an Sl_MWrErr is asserted with Sl_wrDAck. The IPIF2PCI FIFO is flushed upon asserting the PERR Write interrupt. The Detected Parity Error status register bit is set as well.
- If on a burst transfer the initiator latency timer expires, the PLBV46 PCI Bridge terminates the PCI transaction. The PLBV46 PCI Bridge performs retries up to a parameterized number of times as described earlier for the condition of disconnects with and without data. A time-out cannot occur during a single transfer because the PCI32 core requires completion of one data transfer after the latency timer expires.
- If a target abort occurs during either a single or burst write operation, the PLB Master Write Target Abort interrupt is asserted. If a burst write is in progress, Sl_MWrErr is asserted with Sl_wrDAck. Recall that a target abort often indicates that the target cannot proceed with subsequent transactions; this is expected to be a major failure most likely requiring a reset.
- If a PLB write request indicates a burst length that extends beyond the valid range of the IPIF BAR, as defined by the C_IPIF_HIGHADDR_X parameter, the PLB Write Slave BAR Overrun interrupt is asserted. The PLBV46 PCI Bridge does not initiate a write on the PCI bus even though it responds to the PLB Master with Sl_wrDAck.

Table 22 summarizes the abnormal conditions that a PCI target can respond with and how the response is translated to the PLB master.

Table 22: **Response of PLB Master/PCI Initiator write to a remote PCI target with abnormal condition on PCI bus**

Abnormal Condition	Single Transfer	Burst (PLB_wrBurst asserted)
SERR (includes parity error on address phase)	PLB Master Write SERR interrupt asserted	If transfer is in progress, SI_MWrErr is asserted with SI_wrDAck. PLB Master Write SERR interrupt asserted
PLBV46 PCI Bridge Master abort (no PCI target response)	PLB Master Abort Write interrupt asserted	If transfer is in progress, SI_MWrErr is asserted with SI_wrDAck. PLB Master Abort Write interrupt is asserted and the FIFO is flushed.
Target disconnect without data (PCI Retry)	Automatically retried a parameterized number of times. If the last of the PCI write command retries fails due to a PCI Retry, the PLB Master Write Retry interrupt is asserted.	Automatically retried a parameterized number of times. If the last of the PCI write command retries fails due to a PCI Retry, the PLB Master Write Retry interrupt is asserted.
Target disconnect without data (after one completed data phase)	N/A	Automatically retried a parameterized number of times. If the last of the PCI write command retries fails due to a Disconnect with(out) Data, the PLB Master Write Retry Disconnect interrupt is asserted.
Target disconnect with data	Completes	
PERR	Transaction completes and PLB Master Write PERR interrupt asserted	PLB Master Write PERR interrupt asserted. If the burst write is still in progress, SI_MErr is asserted with SI_wrDAck. The FIFO is flushed.
Latency timer expiration	N/A because PCI32 core waits for one transfer after timeout occurs	Automatically retried a parameterized number of times. If the last of the PCI write command retries fails due to a Latency Timer expiration, the PLB Master Burst Write Retry Timeout interrupt is asserted. The PLB master must reissue command per PCI spec if last termination was a retry.
Target Abort	Assert PLB Master Write Target Abort interrupt	Assert PLB Master Write Target Abort interrupt. If the burst write is still in progress, SI_MWrErr is asserted with SI_wrDAck.
Address increments beyond valid range	N/A	Stop PCI transaction. Assert PLB Write Slave BAR Overrun interrupt.

PCI Initiator Initiates a Read Request of a PLB Slave

This section discusses the operation of a remote PCI initiator asserting both single and multiple read commands to read data from a remote PLB slave. For these transactions, the PCI32 core is the PCI target.

Because all PLB address space must be memory space in the PCI sense, memory read, memory read multiple and memory read line are the only read commands from a remote PCI initiator that the PLBV46 PCI Bridge responds to. The I/O read command is ignored and the configuration read command is responded to by the PCI32 core, but has limited impact on the PLBV46 PCI Bridge.

The PLBV46 PCI Bridge translates a PCI memory read multiple command to a PLB burst read. A PCI memory read command that is asserted with multiple data phases requested, such as when FRAME# and IRDY# are asserted on the same clock, is also translated to a PLB burst read. A PCI memory read command that is asserted with a single data phases requested is translated to a PLB single read, such as when FRAME# is deasserted prior to IRDY# being asserted. Table 20 shows translations of PCI commands to PLB transactions.

For PCI memory read commands that are translated to a PLB single read, the address presented on the PCI is translated to the PLB address space by high-order bit substitution with the 2 LSBs set as defined by the byte enable vector for the first data phase. The LSBs are set to the lowest address of the byte lane asserted in the byte enable vector as required by the Xilinx PLB specification. Byte enables from the PCI bus are passed correctly to the PLB in single PLB read transactions. For PCI commands that are translated to a PLB burst read, the address presented on the PLB is word aligned.

Every PCI command that translates to a burst read operation is performed with the full 32 bits on the PLB independent of the byte enable specified by the PCI initiator. The byte enable bits asserted by the PCI initiator in memory read multiple operations of a PLB slave are ignored, and all bytes are read during the PLB burst read operation per PLB protocol. Hence, dynamic byte enable is not supported by PCI initiator burst read from PLB slaves. The system designer must ensure that a burst read with all byte enables asserted is not destructive.

The user must ensure that corrupting the fidelity of the PCI read command with arbitrary byte enables asserted by translating to a PLB burst with all byte enable asserted is not destructive.

Furthermore, it is the responsibility of the PCI initiator to properly read data from non-prefetchable PLB slaves. For example, it must perform single transaction reads of non-prefetchable PLB slaves to avoid destructive read operations of a PLB slave. However, some protection is provided in the hardware as described in a later subsection.

When the PLBV46 PCI Bridge decodes a PCI read command that is for a remote PLB Slave, the transfer is retried on the PCI bus until the requested data has been prefetched from the remote PLB Slave. This is true for both single and burst transactions. Only one PCI initiator read of a PLB slave is supported at a time. After this transaction is successfully completed, a subsequent PCI read with the same PCI command and address then completes on the PCI bus.

A Discard Timer is used to determine how long the PLBV46 PCI Bridge should wait for a subsequent PCI read with the same PCI command and address before discarding the prefetched data in the FIFO.

Data throughput can be very high with burst read transactions. The PCI commands that translate to burst read operations will burst read with a length determined by either a parameterized number or up to the range limit of the PCI BAR, whichever is less. The prefetch read does not read beyond the high-address defined by the PCI BAR length parameter. After the remote PCI initiator terminates the read transaction, the FIFO is flushed of prefetched data that has not been read by the remote PCI initiator.

Abnormal Terminations

- If an address parity error is detected, the PCI32 core will either claim the transaction and issue a Target Abort, or will not claim the transaction and a Master Abort will occur (see PCI32 core documentation). When a Target Abort is issued, the PCI32 core asserts SERR_N, if enabled.
- If SERR_N is asserted by a remote agent in a data phase on either a single or a burst transfer, it is left to the PCI initiator to report the error and initiate any recovery effort that may be needed. The PLBV46 PCI Bridge disconnects with data as soon as possible and any data left is the internal FIFOs are discarded.
- If, on either a single or a burst transfer, a PERR error is detected during a data phase, the PLBV46 PCI Bridge does nothing. Whether the PCI initiator continues or not is initiator dependent.

- If, during a single or a burst transfer prefetch, a PLB rearbitrate is asserted by the PLB slave, the PLBV46 PCI Bridge automatically retries the PLB request until it is successful, or the limit of 2028 retries is reached. If the limit is reached, the PLB Master Read Rearb Timeout interrupt is asserted.
- If a PLB Sl_MRdErr occurs during a single or a burst transfer prefetch, the PCI interrupt is strobed. Sl_MErr can be asserted due to an address phase timeout or a slave assertion of the error signal.
- If, during a burst transfer prefetch, a PLB slave asserts PLB_MRdBTerm which terminates the PLB burst read, the PLBV46 PCI Bridge automatically retries the PLB request and attempts to prefetch the parameterized number of data words or up to the range limit.
- On a burst transfer prefetch, the address will not prefetch beyond the valid range. The IP Master in the bridge attempts to prefetch the parameterized number of data words from addresses up to the limit of the valid range which is defined by the PCIBAR length parameter. All transactions on the PLB will be burst reads of the PLB slave that are terminated by the slave, terminated by the bridge receiving the parameterized number of data words, or terminated when the last address of the defined range is reached. This response is adopted rather than a target abort which is an option per PCI specification. Recall that the PCI32 core cannot throttle data as a target after the first data phase. As data is read by the PCI agent, a disconnect occurs when the FIFO is emptied.

Table 23 summarizes most PLB slave abnormal conditions in a memory read command and how the response is translated to the PCI initiator.

Table 23: **Response to PCI initiator doing a read of a remote PLB slave that terminates the transfer with an abnormal condition on PLB bus**

Abnormal Condition	Memory Read (single)	Memory Read (burst) or Memory Read Multiple
SERR	Target abort by PCI32 core, but completes PLB transaction. Flush FIFOs and assert PLB-side Read SERR interrupt.	Target abort byPCI32 core, but terminates PLB transaction. Flush FIFOs and assert PLB-side PCI Initiator Read SERR interrupt.
PERR	PLBV46 PCI Bridge ignores the signal and continues.	PLBV46 PCI Bridge ignores the signal and continues.
PLB Rearbitrate	Automatically retries PLB read request and, if not success full after 2028 retries, asserts PLB Master Read Rearb Timeout interrupt.	Automatically retries PLB read request and, if not success full after 2028 retries, asserts PLB Master Read Rearb Timeout interrupt.
PLB Sl_MRdErr (including remote slave IPIF timeout)	Assert PCI interrupt	Assert PCI interrupt
PLB PLB_MRdBTerm	N/A	Automatically retries PLB read request and attempts to prefetch all data required.
Address increments beyond valid range	N/A	Disconnect with data on the last valid address on the PCI bus.

PCI Initiator Initiates a Write Request to a PLB Slave

This section discusses the operation of a remote PCI initiator asserting the memory write command to write data to a remote PLB slave. For these transactions, the PCI32 core is the PCI target.

Because all PLB address space must be memory space in the PCI sense, the memory write command is the only write command from a remote PCI initiator to which the PLBV46 PCI Bridge will respond. The command decode and number words written dictates whether the PLB write operation is a burst or single. Byte enables are buffered with data on remote PCI initiator writes to a remote PLB slave, but only transferred for singles because the PLB write protocol does not support dynamic byte enable. All byte enables must be asserted in multiple data phase burst transactions. The command I/O write is ignored and the configuration write command is responded to by the PCI32 core but has limited impact on the PLBV46 PCI Bridge.

All memory write commands are posted, with error notification mostly likely occurring after the PCI transaction with the bridge has completed. The main reason for posted operation is that the PCI32 core does not permit data throttling by the PLBV46 PCI Bridge to utilize PLB burst write commands without buffering data. It is desirable to utilize the PLB burst write command when possible to increase data throughput.

To utilize burst write PLB transactions, data is buffered in the IPIF master PCI2IPIF FIFO until either the PCI write operation terminates or until the PCI2IPIF FIFO is full. If C, the data are burst written over the PLB until the FIFO is emptied, which can take multiple transactions if the PLB slave terminates the transaction. If the PCI write is terminated before the PCI2IPIF FIFO is full, the IPIF master burst writes starts after the PCI transaction ends. The bridge attempts to burst write all the data to the PLB slave device.

Although dynamic byte enable is supported on the PCI bus, dynamic byte enable is not supported by the PLBV46 PCI Bridge because the PLB protocol requires all byte enables to be asserted during burst writes on the PLB. It is the responsibility of the user to ensure that all byte enables be asserted on the PCI in burst write operations to the PLBV46 PCI Bridge.

A PCI initiator can write any number of words of data in a burst operation to the PLBV46 PCI Bridge and the bridge attempts to burst the data to the PLB slave in a burst write operation on the PLB. The slave can terminate the PLB burst or the FIFO can empty because the FIFO is not filled as fast as the data is transmitted over the PLB.

The PLBV46 PCI Bridge can accept a PCI initiator write while a read prefetch is in process. However, only one PCI initiator write to a PLB slave is supported at a time. It is possible for the PLBV46 PCI Bridge to be completing a posted write operation when another write command is received. When this happens, the PLBV46 PCI Bridge forces the PCI to disconnect without data until the posted write operation to a remote PLB slave has completed.

A write to a remote slave that is terminated before the FIFO is emptied is automatically retried by the PLB/PCI bridge. Address bookkeeping is performed in the IPIF to permit the correct sequence of PLB transactions as either bursts or single transactions and/or combinations of the two as required to complete the transfer.

Abnormal Terminations

- If an address parity error is detected, the PCI32 core will either claim the transaction and issue a target abort, or will not claim the transaction and a master abort will occur (see PCI32 core documentation). If enabled, the PCI32 core asserts SERR_N when address phase parity errors are detected.
- If SERR_N is asserted by a remote agent in a data phase, the bridge disconnects without data for burst transfers and the PLB-side PCI Initiator Write SERR interrupt is asserted. If the SERR occurs after the IP master device has started a PLB transaction, the PLB transaction is terminated as soon as possible. The PLBV46 PCI Bridge flushes any data and resets for a subsequent transaction. It is left to the PCI initiator to report the error on the PCI-side and initiate any recovery effort that may be needed.
- If a PERR error is detected on a write transfer, the PCI32 core asserts the PERR signal, if enabled, and sets the Detected PERR error in the status register. The PLBV46 PCI Bridge disconnects without data for burst transfers. On the PLB-side, the bridge terminates the PLB transfer as soon as possible if the transaction is in progress. Due to the latency in PERR, the data for which the PERR was detected most likely has been written to the PLB slave. It is left to the PCI initiator to report the error and initiate any recovery effort that may be needed.
- If at any time while data from the PCI2PLB_FIFO is being written to a PLB slave, a PLB rearbitrate occurs, the PLBV46 PCI Bridge performs write retries until successful, or the limit of 2028 retries is reached. If the limit is reached, the PLB Master Write Rearb Timeout interrupt is asserted. The PLBV46 PCI Bridge IP master write state machine is tied up during the retry operation, therefore, PCI initiator writes are inhibited. Target disconnects without data (PCI retry) are asserted for subsequent PCI transactions when the transactions are inhibited.

- If during a write command a PLB slave asserts PLB_MWrBTerm which terminates the PLB burst write, the PLBV46 PCI Bridge automatically retries the PLB request and attempts to empty the fifo. The behavior is the same as that described for the PLB rearbitrate previously.
- If a PLB Sl_MWrErr occurs while data from the write buffer is being written to a PLB slave, the IP Master will abort the PLB transaction. When this occurs, the PLBV46 PCI Bridge strobes the PCI interrupt. Sl_MWrErr can be asserted due to an address phase timeout or a slave assertion of the error signal. Data in the write buffer is flushed when the PCI interrupt is strobed.
- If on a write command transaction the PCI initiator attempts to go beyond the valid address range, the PLBV46 PCI Bridge does not accept data beyond the valid range. Only valid data is buffered in the bridge and all buffered data is transferred to the PLB slave. This is adopted rather than a target abort. Due to pipelining in the PCI32 core, disconnect without data can occur if the initiator is throttling the data when the first address is near the end of the valid range.

Table 24 summarizes most abnormal conditions that a PLB slave can respond with to a memory write command and how the response is translated to the PCI initiator.

Table 24: **Response to PCI initiator doing a write to a remote PLB slave that terminates the transfer with an abnormal condition on a bus**

Abnormal Condition	Memory Write
Parity Error on Address phase	PCI32 core dictates response with target abort or not accepting transaction. SERR_N is asserted if enabled
SERR on data phase	Disconnect with data for burst transfers and assert PLB-side PCI Initiator Write SERR interrupt
PERR on data phase	Disconnect with data for burst transfers and terminate PLB transfer
PLB Rearbitrate	Automatically retried and, if not success full after 2028 retries, asserts PLB Master Write Rearb Timeout interrupt.
PLB Sl_MWrErr	Disconnect with data if PCI transfer is in progress, flush FIFO, and strobed the PCI interrupt
PLB_MWrBTerm asserted	Automatically retried until successful.
Address increments beyond valid range	Accept data from only valid address on the PCI bus. Disconnect to terminate the PCI transaction.

Configuration Transactions

Functionality for host bridge configuration of PCI agents can be implemented in the PLBV46 PCI Bridge at build time by setting C_INCLUDE_PCI_CONFIG=1. When the bridge is not configured with host bridge configuration functionality, IDSEL of the PCI32 core is connected to the IDSEL port of the bridge. When the bridge is configured with host bridge configuration functionality, IDSEL of the PCI32 core is connected internally to the specified address signal (as described in the next paragraphs) and the IDSEL port of the bridge is not used. As with Memory and I/O data transactions, byte addressing integrity is maintained in configuration transfers across the bus.

When host bridge configuration functionality is implemented in the PLBV46 PCI Bridge, the PCI32 core in the PLBV46 PCI Bridge must be configured first. The minimum that must be set is the Bus master enable bit in the command register and the latency timer register. This requirement is because the PCI32 core has the capability to configure only itself until the Bus master enable bit is set in the command register of the PCI32 core and the latency timer register is properly set to avoid timeouts. If the PCI32 core latency timer is set to 0 value, configuration writes to remote PCI devices do not complete and configuration reads of remote PCI devices will terminate due to the latency timer expiration. Configuration reads of remote PCI devices with the latency timer set to 0 will return `0xFFFFFFFF`.

Table 25 shows the results of configuring the PCI32 core configuration header in the PLBV46 PCI Bridge by both PLB-side configuration transactions and by remote PCI host bridge configuration transactions from the PCI-side. This example assumes all PCI BARs are designated memory space which is the only allowed PCIBAR memory type. The PLB-side configuration of the PCI32 core enables all functionality in the Command Status Register and sets the latency timer to maximum count for most any data value written to the registers. This behavior is an artifact of the v3.0 PCI32 core used in Spartan®-3 and Virtex-4 devices. However, the v4.0 PCI32 core used in the Virtex-5 device family DOES NOT exhibit this behavior.

Configuration Space Header

The PCI32 core used in the PLBV46 PCI Bridge can be configured with functionality to address a wide range of applications.

Fields of the Configuration Space Header are Device ID, Vendor ID, Class Code, Rev ID, Subsystem ID, Subsystem Vendor ID, Maximum Latency and Minimum Grant. The parameters for these fields are C_DEVICE_ID, C_VENDOR_ID, C_CLASS_CODE, C_REV_ID, C_SUBSYSTEM_ID, C_SUBSYSTEM_VENDOR_ID, C_MAX_LAT, C_MIN_GNT, respectively.

Listed in Table 25 are details on the remaining configuration registers that are fixed in value.

BIST, Line Size and Expansion ROM Base Address are not implemented in the PCI32 design.

Header Type is a fixed byte of all zeros in the PCI32 design.

Cardbus CIS Pointer is set to all zeros for the PCI32 implementation used in the PLBV46 PCI Bridge.

Capabilities Pointer is not enabled for the PCI32 implementation used in the PLBV46 PCI Bridge.

Interrupt Pin register is set to `0x01`.

BAR3, BAR4 and BAR5 are not supported by the PCI32 Core. For these registers and un-implemented PCIBARs (determined by C_PCIBAR_NUM), zeros are returned when read. Writes to the un-implemented configuration space addresses have no effect.

Latency timer, BAR0, BAR1, and BAR2 are required to be set by the host bridge as necessary. The number of BARs (0-3) is set by the parameter C_PCIBAR_NUM.

The User Configuration Space is enabled for the PCI32 implementation used in the PLBV46 PCI Bridge.

Table 25: **Results of PCI32 core Command Register configuration by remote host bridge (PCI-side) and by self-configuration (PLB-side) Note: Results for Virtex-5 FPGA self-configuration is the same as remote host bridge configuration**

	Results in Command Register after write (PLB-side byte swapped format)	
Data Written (PLB-side byte swapped format)	**by remote host bridge (Virtex-4, Virtex-5 and Spartan-3) and by self-configuration (Virtex-5)**	**by self-configuration (Virtex-4 and Spartan-3)**
0x0000	0x0000	0x4605
0x0100	0x0000	0x4605
0x0200	0x0200	0x4605
0x0300	0x0200	0x4605
0x0400	0x0400	0x4605
0x0500	0x0400	0x4605
0x8600	0x0600	0x4605
0x8700	0x0600	0x4605
0xFFFF	0x4605	0x4605

Note:

1. This assumes that the PCI BARs in the PCI32 core are configured to only Memory type and not I/O-type which is not an allowed configuration. After self-configuration, a remote initiator can reconfigure the PCI32 core to any valid state.

Table 26: **Results of PCI32 core Latency Timer Register configuration by remote host bridge (PCI-side) and by self-configuration (PLB-side) Note: Results for Virtex-5 FPGA self-configuration same as remote host bridge configuration**

	Results in Latency Timer Register after write (PLB-side byte swapped format)	
Data Written	**by remote host bridge (Virtex-4, Virtex-5 and Spartan-3) and by self-configuration (Virtex-5)**	**by self-configuration (Virtex-4 and Spartan-3)**
0x00	0x00	0xFF
0x01	0x01	0xFF
0xFF	0xFF	0xFF

Table 25 and Table 26 show examples only and do not show all the possible bit patterns. The bytes are swapped for maintaining byte addressing integrity.

The PCI32 core is PCI 2.2 compliant core, but it has PCI 2.3 compliant features. The PCI32 core documentation should be reviewed for details of compliance.

Configuration transactions from the PLB-side of the bridge are supported by the PLBV46 PCI Bridge. The protocol follows the PCI 2.2 specification but with changes required to adapt to the PLB-side bus protocol. The primary difference is that all registers (Configuration Address Port, Configuration Data Port, and Bus Number/Subordinate Bus Number) are on the PLB-side of the bridge and are not accessible from the PCI-side via I/O transactions on the PCI bus. This approach is adopted so that one BAR of the PCI32 core is not required for the Configuration Port registers. The registers are mapped relative to the bridge device base address as shown in Table 5. The registers exist only if the bridge is configured with PCI host bridge configuration functionality.

Data is loaded in the Configuration Address Port with the Byte format specified in the PCI 2.2. specification. A PLB-side read of the Configuration Data Port initiates a Configuration Read command with data returned to the PLB-side upon completion of the PCI-side read command. A PLB-side write to the Configuration Data Port register initiates a Configuration Write transaction on the PCI bus. Determination of whether the read or write transfer is type 0 or type 1 is done automatically.

Both type 0 and type 1 configuration transactions are supported. The type of transaction is determined from the Bus number in the Configuration Address Port register (Bits 8-15) and the bus numbers in the Bus Number/Subordinate Bus Number register. The local bus number is located at bits 8-15 and the maximum subordinate bus number is located at bits 24-31 in the Bus Number/Subordinate Bus Number register. If the Bus number in the Configuration Address Port register is equal to the local bus number in the Bus Number/Subordinate Bus Number register (bits 8-15), a type 0 transaction is performed. If the Bus number in the Configuration Address Port register is greater than the bus number in the Bus Number/Subordinate Bus Number register and less than or equal to the maximum subordinate Bus number, a type 1 transaction is performed. If a configuration transaction to a Bus Number not satisfying the inequality relation is attempted, then PLB Sl_MErr is asserted. When a configuration read from a bus number not in the subordinate bus range is initiated, nothing occurs on the PCI bus and the PLB Sl_MErr signal is asserted. When a configuration write to a bus number not in the subordinate bus range is initiated, nothing occurs on the PCI bus, the data is discarded and PLB Sl_MErr is asserted. These conditions are equivalent to the situation where the master enable bit in the configuration command register of the PCI32 core is not set.

If a configuration read to a device number not assigned to a device on the PCI bus is attempted, a Master Abort occurs on the PCI bus, and all ones are returned on the PLB bus.

IDSEL is asserted for the device to be configured in all type 0 configuration transactions. The most common implementation method for IDSEL is used in this bridge implementation where address lines AD[31:16] are required to be mapped to IDSEL for each device.

The mapping is:.

- IDSEL of device 0 is connected to AD16
- IDSEL of device 1 is connected to AD17
- IDSEL of device 2 is connected to AD18.
- ...
- IDSEL of device 15 is connected to AD31

A decode of the device number in the Configuration Address Port is used to determine which address line/IDSEL is asserted.

As noted, when the bridge has host bridge configuration functionality, IDSEL of the PCI32 core is connected internally to the AD-bit specified by the C_BRIDGE_IDSEL_ADDR_BIT parameter.

C_NUM_IDSEL specifies the number of PCI agents that can be configured on the PCI bus by specifying the number of IDSEL lines that are decoded and assigned to address lines AD[31:16]. Each device on the bus must have its IDSEL line properly connected to the PCI AD bus. It can be resistively-coupled to the associated address bit or direct coupling, if it is not detrimental to performance per PCI 2.2 specification. Because the PCI32 core does not support address stepping, resistive coupling of IDSEL with the assigned address bit must be sufficient to ensure proper signal levels at IDSEL without utilizing address stepping.

Multiple PLBV46 PCI Bridges can be instantiated on a given PLB. Each bridge has a unique base address with fixed offset to corresponding unique set of configuration registers. The unique set of configuration registers are used to perform configuration accesses on the unique primary PCI bus and its' subordinate buses. Device numbers are independent for each PLBV46 PCI Bridge instantiated, but bus numbering must be monotonically increasing for all primary buses and their subordinate buses.

Abnormal Terminations

Responses to abnormal terminations of Configuration Reads and Writes follow closely to single reads and writes by a remote PLB master from or to a remote PCI target. Details of each transaction can be reviewed in the previous sections; however, some differences exist. Shown in Table 27 is a table summary of responses to abnormal terminations during configuration transactions. The differences as compared to PLB master read and writes to remote targets are shown.

Table 27: **Response of PLB Master/v3.0 Initiator Configuration Transactions with abnormal condition on PCI bus**

Abnormal Condition	Configuration Read	Configuration Write
SERR (including address phase parity error)	Return all ones and set PLB Master Read SERR interrupt	PLB Master Write SERR interrupt asserted
PLBV46 PCI Bridge Master abort (no PCI target response)	All 1s are returned	PLB Master Abort Write interrupt asserted
Target disconnect without data (PCI Retry)	Automatically retried until the transfer completes	Automatically retried a parameterized number of times. If the last of the PCI write command retries fail due to a PCI retry, the PLB Master Burst Write Retry interrupt is asserted. The PLB master must reissue command per PCI specification, if last termination was a retry.
Target disconnect with data	Completes	Completes
PERR	Data is transferred and PLB Master Read PERR interrupt is asserted	Transaction completes and PLB Master Write PERR interrupt asserted
Latency timer expiration Latency timer register must be set to non-zero value for accessing remote devices.	N/A because PCI32 core waits for one transfer after timeout occurs when latency timer is non-zero	N/A because PCI32 core waits for one transfer after timeout occurs when latency timer is non-zero
Target Abort	Set PLB Master Read Target Abort interrupt and terminate PLB transaction with Slv_MErr assertion	Assert PLB Master Write Target Abort interrupt.

Design Implementation

Design Tools

The PLBV46 PCI Bridge design is implemented using the VHDL. All coding standards and abbreviations specified in *IPSPEC001 Virtex-II Pro Coding Standards* and *IPSPEC002 Virtex-II Pro Standard Abbreviations* have been adhered to.

Xilinx XST and Synplicity Synplify Pro synthesis tools are used for synthesizing the PLBV46 PCI Bridge. The NGC format from XST and EDIF netlist output from Synplify Pro are then input to the tool suite for actual device implementation.

Design Debug

The PLBV46 PCI Bridge has a test vector output (PCI_monitor) to facilitate system debug, such as when adding an ILA to a system. The test vector allows monitoring the PCI bus and is the output of IO-buffers that are instantiated in the PCI32 core. PCLK, RCLK, and Bus2PCI_INTR are not included in the test vector because these signals do not have io-buffers instantiated in the core and are accessible to use directly at the core top-level or above. If the port is not connected in the EDK tool top-level mhs-file, the wrapper leaves this port open. PCI Bus monitoring test vector bit definition is listed in Table 28.

Table 28: **PCI Bus Monitoring Signals**

Bit Index	Signal Name	Instantiated I/O-Buffer
PCI Transaction Control Signals		
0	FRAME_N	Yes
1	DEVSEL_N	Yes
2	TRDY_N	Yes
3	IRDY_N	Yes
4	STOP_N	Yes
5	IDSEL	Yes
PCI Interrupt Signals		
6	INTR_A	Optional
PCI Error Signals		
7	PERR_N	Yes
8	SERR_N	Yes
PCI Arbitration Signals		
9	REQ_N	Optional
10	reserved	NA
PCI Address, Datapath, and Command Signals		
11	PAR	Yes
12-43	AD[31:0]	Yes
44-47	CBE[3:0]	Yes

Design Verification

The PLBV46 PCI Bridge design is verified according to IPSPEC000 PLBV46 PCI Bridge Verification Plan.

Design Contraints

The PLBV46 PCI Bridge uses the PCI32 core that requires specific constraints to meet PCI specifications. UCF-files with the constraints for the PCI32 core in many different packages are available from the LogiCORE IP Lounge. The PCI32 core specific constraints can be included in the top-level UCF by the user.

The constraints are also implemented automatically in the EDK tool flow with any tool option that invokes bridge synthesis. In this flow, TCL-scripts generate the UCF constraints and place them in a file in the PLBV46 PCI Bridge directory of the project implementation directory. The UCF constraints are then included in the NGC file generated in the EDK tool flow. The user can check the UCF in the implementation directory of the bridge directory to verify that the constraints are included. As noted earlier, the user can include all constraints in the top-level UCF.

When the constraints are included in both the top-level UCF and the bridge NGC file (via the bridge directory UCF), then the top-level UCF overrides any conflicting constraints in the bridge NGC file.

To remind the user that the following constraints must be included, PLATGEN generates the message:

```
The PLBV46 PCI Bridge design requires design constraints to guarantee performance. Please
refer to the PLBV46 IPIF/LogiCORE PCI bridge design data sheet for details.
```

Additional bridge specific constraints are required and an example UCF is provided in the EDK pcores library. To remind the user that the additional bridge related constraints must be included in the top-level UCF, PLATGEN generates the message:

```
An example UCF is available for this core and must be modified for use in the system. Please
refer to the EDK Getting Started guide for the location of this file.
```

The constraints that the PCI32 core require to meet PCI specifications are shown in the following constraints.

All I/O buffers must have IOB=TRUE

IOSTANDARD must explicitly list PCI33_3. Both BYPASS IOBDELAY=BOTH must be included for all PIC ports, as shown in these examples.

```
NET "PCI_AD(*)"    IOSTANDARD=PCI33_3;
NET "PCI_CBE(*)"   IOSTANDARD=PCI33_3;
NET "PCI_PAR"      IOSTANDARD=PCI33_3;
NET "PCI_FRAME_N"  IOSTANDARD=PCI33_3;
NET "PCI_TRDY_N"   IOSTANDARD=PCI33_3;
NET "PCI_IRDY_N"   IOSTANDARD=PCI33_3;
NET "PCI_STOP_N"   IOSTANDARD=PCI33_3;
NET "PCI_DEVSEL_N" IOSTANDARD=PCI33_3;
NET "PCI_PERR_N"   IOSTANDARD=PCI33_3;
NET "PCI_SERR_N"   IOSTANDARD=PCI33_3;
#Include next 2 if routed to pins
NET "IDSEL" IOSTANDARD=PCI33_3;
NET "GNT_N"   IOSTANDARD=PCI33_3;

NET "PCI_AD(*)"    BYPASS;
NET "PCI_CBE(*)"   BYPASS;
NET "PCI_PAR"      BYPASS;
NET "PCI_FRAME_N"  BYPASS;
NET "PCI_TRDY_N"   BYPASS;
NET "PCI_IRDY_N"   BYPASS;
NET "PCI_STOP_N"   BYPASS;
NET "PCI_DEVSEL_N" BYPASS;
NET "PCI_PERR_N"   BYPASS;
NET "PCI_SERR_N"   BYPASS;
#
```

```
NET "*/RST_N"      IOBDELAY = BOTH ;
NET "*/AD<*>"      IOBDELAY = BOTH ;
NET "*/CBE<*>"     IOBDELAY = BOTH ;
NET "*/REQ_N"      IOBDELAY = BOTH ;
NET "*/GNT_N"      IOBDELAY = BOTH ;
NET "*/PAR"        IOBDELAY = BOTH ;
NET "*/IDSEL"      IOBDELAY = BOTH ;
NET "*/FRAME_N"    IOBDELAY = BOTH ;
NET "*/IRDY_N"     IOBDELAY = BOTH ;
NET "*/TRDY_N"     IOBDELAY = BOTH ;
NET "*/DEVSEL_N"   IOBDELAY = BOTH ;
NET "*/STOP_N"     IOBDELAY = BOTH ;
NET "*/PERR_N"     IOBDELAY = BOTH ;
NET "*/SERR_N"     IOBDELAY = BOTH ;
NET "*/PCI_INTA"   IOBDELAY = BOTH ;
```

TNM constraints must be defined as specified in the *PCI32 Design Guide* and PCI32 core UCFs. These parameters are automatically set in the normal EDK tool flow, but can be included in the system top-level UCF. For alternative tool flows, the settings are shown in the following Time Specs example. When the complete set of constraints is used, the PCI clock must be a PAD input which is the required clock routing for all PCI32 core implementations. The EDK flow checks if the PCI clock is a PAD input and if it is, then the OFFSET constraints shown in the following Time Specs example are included in the bridge NGC file.

```
##############################################################################
# Time Specs
##############################################################################
#
# Important Note: The timespecs used in this section cover all possible
# paths. Depending on the design options, some of the timespecs might
# not contain any paths. Such timespecs are ignored by PAR and TRCE.
#
#             1) Clock to Output       =     11.000 ns
#             2) Setup                 =      7.000 ns
#             3) Grant Setup           =     10.000 ns
#             4) Datapath Tristate     =     28.000 ns
#             5) Period                =     30.000 ns
#
# Note: Timespecs are derived from the PCI Bus Specification. Use of
# offset constraints allows the timing tools to automatically include
# the clock delay estimates. These constraints are for 33 MHz operation.
#
# The following timespecs are for setup.
#
TIMEGRP "PCI_PADS_D" OFFSET=IN   7.000 VALID  7.000 BEFORE "PCI_CLK" TIMEGRP "ALL_FFS"  ;
TIMEGRP "PCI_PADS_B" OFFSET=IN   7.000 VALID  7.000 BEFORE "PCI_CLK" TIMEGRP "ALL_FFS"  ;
TIMEGRP "PCI_PADS_P" OFFSET=IN   7.000 VALID  7.000 BEFORE "PCI_CLK" TIMEGRP "ALL_FFS"  ;
TIMEGRP "PCI_PADS_C" OFFSET=IN   7.000 VALID  7.000 BEFORE "PCI_CLK" TIMEGRP "ALL_FFS"  ;
#
# The following timespecs are for clock to out where stepping is not used.
#
TIMEGRP "PCI_PADS_D" OFFSET=OUT 11.000 AFTER  "PCI_CLK" TIMEGRP "FAST_FFS" ;
TIMEGRP "PCI_PADS_B" OFFSET=OUT 11.000 AFTER  "PCI_CLK" TIMEGRP "FAST_FFS" ;
TIMEGRP "PCI_PADS_P" OFFSET=OUT 11.000 AFTER  "PCI_CLK" TIMEGRP "FAST_FFS" ;
TIMEGRP "PCI_PADS_C" OFFSET=OUT 11.000 AFTER  "PCI_CLK" TIMEGRP "ALL_FFS"  ;
```

```
#
# The following timespecs are for clock to out where stepping is used.
#
TIMEGRP "PCI_PADS_D" OFFSET=OUT 28.000 AFTER  "PCI_CLK" TIMEGRP "SLOW_FFS" ;
TIMEGRP "PCI_PADS_B" OFFSET=OUT 28.000 AFTER  "PCI_CLK" TIMEGRP "SLOW_FFS" ;
TIMEGRP "PCI_PADS_P" OFFSET=OUT 28.000 AFTER  "PCI_CLK" TIMEGRP "SLOW_FFS" ;
```

Target Technology

The intended target technology is for the Spartan-3, Spartan-6, Virtex-4, and Virtex-5 FPGAs.

Virtex-4 and Virtex-5 FPGA Support

To meet PCI specification setup and hold times with the Virtex-4 and Virtex-5 architectures, it is necessary to insert an IDELAY primitive between the pad and I/O buffer of most PCI signals and to include additional constraints in the UCF. When IDELAY primitives are used in the mode required by the PCI32 core, IDELAYCTRL (idelay controllers) are required. Also required is a 200 MHz reference clock supplied by the user which is used by both IDELAY and IDELAYCTRL primitives. These primitives are only required for Virtex-4 and Virtex-5 architectures. The additional constraints are discussed after the discussion of primitives specific to Virtex-4 and Virtex-5 devices.

The 200 MHz clock is input to port RCLK and must be driven by a global buffer. If the architecture is not off the Virtex-4 or Virtex-5 devices, the port does not connect to anything in the plbv46_pci bridge, and it might be omitted from the MHS-file. This allows upgrading to v1.02.a from v1.01.a without changing ports. Recall that v1.01.a does not support the Virtex-4 architecture. It is required that the 200 MHz clock be stable when PLB_RST is asserted to the PLBV46 PCI Bridge. An unstable clock can result failure of PLBV46 PCI Bridge operation. The clock source can be an external source or generated with a DCM in the FPGA. Application Notes and Implementation Guides for the PCI32 core, as well as reference designs using the PLBV46 PCI Bridge, present options for generating the 200 MHz clock.

IDELAY primitives are instantiated automatically by the bridge when the C_FAMILY parameter is set to the Virtex-4 or Virtex-5 architecture. The EDK tools automatically set this parameter and it cannot be changed by the user. There is a special case to consider for instantiation of IDELAY primitives. Port GNT_N requires the IDELAY primitive only if the port is connected to a package pin. If GNT_N is connected to an internal signal (an FPGA internal arbiter such as pci_arbiter_v1_00_a) or connected to ground, then an IDELAY primitive is not needed. EDK tools have the system level information to determine if GNT_N is connected to a pad or has an internal connection. This accomplished with a TCL-script in the PLBV46 PCI Bridge pcore library that is called by the EDK tools.

EDK tools automatically sets the parameter C_INCLUDE_GNT_DELAY which controls if an IDELAY primitive is included in the GNT_N signal path. C_INCLUDE_GNT_DELAY defaults to exclude the IDELAY primitive and must be set by the user if the core is used outside EDK tools with GNT_N connected to a pin.

IDELAYCTRL primitives are not as automatic in the build procedure. It is required that the user instantiate the number of IDELAYCTRL primitive needed for their design and to provide LOC contraints for each IDELAYCTRL. This is required for EDK 8.1 tools because when instantiating only one IDELAYCTRl without LOC constraints, the tools will replicate the primitive throughout the design. Replicating the primitive has the undesirable results of higher power consumption, higher power consumption, utilization of more global clock resources, and greater use of routing resources. To prevent these undesirable results, a procedure is described in the next paragraph for instantiating the IDELAYCTRLs. See the *Virtex-4 FPGA User Guide* discussion of IDELAYCTRL usage and design guidance for more details on IDELAYCTRL and usage. Tools beyond ISE® software 7.1 might handle IDELAYCTRL instantiation differently.

It turns out that the number of signals in the PCI protocol requires at least two IDELAYCTRL primitives when implemented in the Virtex-4 or Virtex-5 architecture. The actual number depends on the pinout defined by the user. To avoid the undesirable results noted previously, the PCI32 core stand-alone core is fixed to use two IDELAYCTRL instantiations and prescribes pinouts that require only two IDELAYCTRL primitives. To provide more flexibility to the user, the PLBV46 PCI Bridge allows specifying the number of IDELAYCTRL primitives from two to six; this is set at build time by set the parameter C_NUM_IDELAYCTRL.

However, it might be difficult to meet timing when the pinout is spread out to require four to six IDELAYCTRL primitives and it is recommended to use a PCI pinout packed together enough to require only two IDELAYCTRL primitives. See the *Virtex-4 User Guide* discussion of IDELAYCTRL usage and design guidance or the *Virtex-4 Library Guide* for IDELAYCTRL primitives for more details.

When more than one IDELAYCTRL is instantiated, the ISE 8.1 tools require LOC constraints on each IDELAYCTRL instantiation. A failure in MAP occurs if the LOC constraints are not provided. The FPGA Editor tool can be helpful to determine IDELAYCTRL LOC coordinates for the user's pinout. The syntax for the UCF LOC constraints is shown in the following example where the instance name in the PLBV46 PCI Bridge for each IDELAYCTRL is XPCI_IDC0 to XPCI_IDCN where N is the C_NUM_IDELAYCTRL-1. The user need only include an LOC entry for each instance used in the system design and not for all possible six IDELAY controllers. For each entry, include the LOC coordinates for the part and pinout in the design. The following example is for a design that uses 2 IDELAYCTRL primitives.

This approach allows users to use the constraint LOC coordinates directly from the PCI32 core ucf-generator. The UCF generator prescribes I/O pin layout that only uses two IDELAYCTRL primitives. The following example is for a system with two IDELAYCTRL primitives with example only coordinates. Depending on the user's pinout, more IDELAYCTRLs might be needed.

```
INST *XPCI_IDC0 LOC=IDELAYCTRL_X2Y5;
INST *XPCI_IDC1 LOC=IDELAYCTRL_X2Y6;
```

An optional method for setting of LOC constraints is to use the C_IDELAYCTRL_LOC parameter. This parameter, when properly set, generates constraints in the bridge core UCF that is combined with the plbv46_pci bridge NGC file during normal EDK tool flow. If the LOC constraints are set in the system top-level UCF, this parameter has no effect for either case of it being properly set or set to default (NOT_SET).

This is because the system top-level UCF overrides all core level ucf constraints. However, if it is not set, then a warning that it is not set is asserted early in the EDK tool flow for the tool options, **generate netlist**, **generate bitstream**, and other tool options that would invoke synthesis of the plbv46_pci bridge.

If the system top-level UCF does include the LOC constraints, then this warning can be ignored. With EDK 8.1 tools, MAP will fail if the LOC coordinates are not provided by at least one of the methods. An example of the syntax for the C_IDELAYCTRL_LOC parameter is shown in the example that follows.

The parameter C_IDELAYCTRL_LOC has the syntax of IDELAYCTRL_XNYM where N and M are coordinates and multiple entries are concatenated by "-" (dash). The order of entries correspond to IDELAYCNTRL instance names XPCI_IDC0, XPCI_IDC1, ... up to the maximum index of IDELAY controller instances in the user's board design. The maximum index is C_NUM_IDELAYCTRL-1. To use the parameter to set the LOC constraint in the core level UCF for the above example, the parameter should be set in the MHS-file as shown this example.

```
PARAMETER C_IDELAYCTRL_LOC="IDELAYCTRL_X2Y5-IDELAYCTRL_X2Y6"
```

The quotes are optional. The actual number of IDELAYCTRL primitives and corresponding LOC constraints depends on the user's PCI pinout and part used.

Other constraints that are required include the IOBDELAY_TYPE, IOBDELAY_VALUE and IOB. These parameters are set in the normal EDK tool flow, but can be included in the system top-level UCF. For alternative tool flows, the setting are shown in the Virtex-4 Only Constraints example. The settings shown are settings at the time this document was written. The *LogiCORE IP v3 PCI32 core Implementation Guide* and v3.0 core ucf generator tool should be checked for updated values. IOSTANDARD must be explicitly defined in the UCF with the BYPASS constraint for ISE 8.1 tools; this can change in with future versions of the tools.

```
#------------------------------------------------------------------------
# Virtex-4 Only Constraints
#------------------------------------------------------------------------
INST "*XPCI_CBD*"                IOBDELAY_TYPE=VARIABLE ;
INST "*XPCI_ADD*"               IOBDELAY_TYPE=VARIABLE ;
INST "*PCI_CORE/XPCI_PARD"       IOBDELAY_TYPE=VARIABLE ;
INST "*PCI_CORE/XPCI_FRAMED"     IOBDELAY_TYPE=VARIABLE ;
INST "*PCI_CORE/XPCI_TRDYD"      IOBDELAY_TYPE=VARIABLE ;
INST "*PCI_CORE/XPCI_IRDYD"      IOBDELAY_TYPE=VARIABLE ;
INST "*PCI_CORE/XPCI_STOPD"      IOBDELAY_TYPE=VARIABLE ;
INST "*PCI_CORE/XPCI_DEVSELD"    IOBDELAY_TYPE=VARIABLE ;
INST "*PCI_CORE/XPCI_PERRD"      IOBDELAY_TYPE=VARIABLE ;
INST "*PCI_CORE/XPCI_SERRD"      IOBDELAY_TYPE=VARIABLE ;
#Include next 2 if routed to pins
INST "*XPCI_IDSEL"               IOBDELAY_TYPE=VARIABLE ;
INST "*XPCI_GNTD"                IOBDELAY_TYPE=VARIABLE ;

INST "*XPCI_CBD*"                IOBDELAY_VALUE=55 ;
INST "*XPCI_ADD*"               IOBDELAY_VALUE=55 ;
INST "*PCI_CORE/XPCI_PARD"       IOBDELAY_VALUE=55 ;
INST "*PCI_CORE/XPCI_FRAMED"     IOBDELAY_VALUE=55 ;
INST "*PCI_CORE/XPCI_TRDYD"      IOBDELAY_VALUE=55 ;
INST "*PCI_CORE/XPCI_IRDYD"      IOBDELAY_VALUE=55 ;
INST "*PCI_CORE/XPCI_STOPD"      IOBDELAY_VALUE=55 ;
INST "*PCI_CORE/XPCI_DEVSELD"    IOBDELAY_VALUE=55 ;
INST "*PCI_CORE/XPCI_PERRD"      IOBDELAY_VALUE=55 ;
INST "*PCI_CORE/XPCI_SERRD"      IOBDELAY_VALUE=55 ;
#Include next 2 if routed to pins
INST "*XPCI_IDSEL"               IOBDELAY_VALUE=55 ;
INST "*XPCI_GNTD"                IOBDELAY_VALUE=55 ;
```

Some of the Virtex-4 FPGA constraints are implemented automatically in the EDK tool flow with any tool option that invokes bridge synthesis. As described earlier, TCL scripts generate the UCF constraints and place them in a file in the PLBV46 PCI Bridge directory of the project implementation directory. The UCF constraints are then included in the NGC file generated in the EDK tool flow. The user can check the UCF in the implementation directory of the bridge directory to verify that the constraints are included. Alternatively, the user can include all constraints in the top-level UCF. When the constraints are included in both the top-level UCF and the bridge NGC file (via the bridge directory UCF), then the top-level UCF overrides any conflicting constraints in the bridge NGC file.

Device Utilization and Performance Benchmarks

Core Performance

Because the PLBV46 PCI Bridge is a module that is used with other design pieces in the FPGA, the utilization and timing numbers reported in this section are just estimates. As the PLBV46 PCI Bridge is combined with other pieces of the FPGA design, the utilization of FPGA resources and timing of the PLBV46 PCI Bridge design varies from the results reported here.

To analyze the PLBV46 PCI Bridge timing within the FPGA, a design was created that instantiated the PLBV46 PCI Bridge with the parameters set as outlined in Table 29. The data is shown for Virtex-4 and Virtex-5 devices.

Table 29: **PLBV46 PCI Bridge FPGA Performance and Resource Utilization Benchmarks**

	Parameter Values				Device Resources					f_{MAX}
Configuration **Description**	C_IPIFBAR_NUM	C_PCI_BAR_NUM	C_IPIF2PCI_FIFO_ABUS_WIDTH C_PCI2IPIF_FIFO_ABUS_WIDTH	C_INCLUDE_PCI_CONFIG	Slices	Slice Flip- Flops	4- input LUTs	# RAMB16s	# GCLK	MHz
Total (with BarOffset and DevNumregs)	6	3	9	1	3188	2753	4015	4	5	
Total (with BarOffset and DevNumregs)	6	3	7	1	3088	2647	3897	4	5	
Total (without BarOffset and DevNum regs)	6	3	9	1	2892	2504	3660	4	5	
Total (without BarOffset and DevNum regs)	6	3	7	1	2801	2398	3544	4	5	
Total (with BarOffset and DevNum regs)	4	2	9	1	3097	2658	3944	4	5	
Total (without BarOffset and DevNum regs)	4	2	9	0	2749	2383	3499	4	5	

Note:

1. These benchmark designs contain only the PLBV46 PCI Bridge with registered inputs/outputs with any additional logic. Benchmark numbers approach the performance ceiling rather that representing performance under typical user conditions.

System Performance

To measure the system performance (Fmax) of this core, this core was added to a Virtex-4 FPGA system, a Virtex-5 FPGA system, and a Spartan-3A FPGA DSP system as the Device Under Test (DUT) as shown in Figure 3, Figure 4, and Figure 5.

Because the PLBv46 PCI Bridge core is 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 designs in the system, the utilization of FPGA resources and timing of the core design varies from the results reported here.

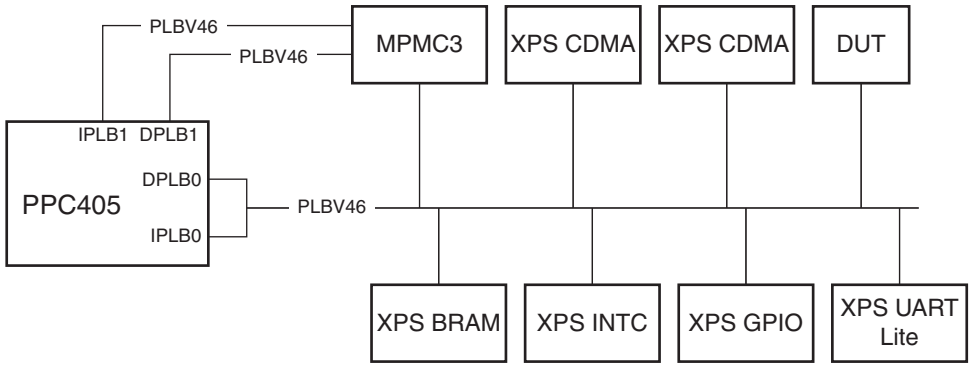

Figure 3: **Virtex-4 FX FPGA System**

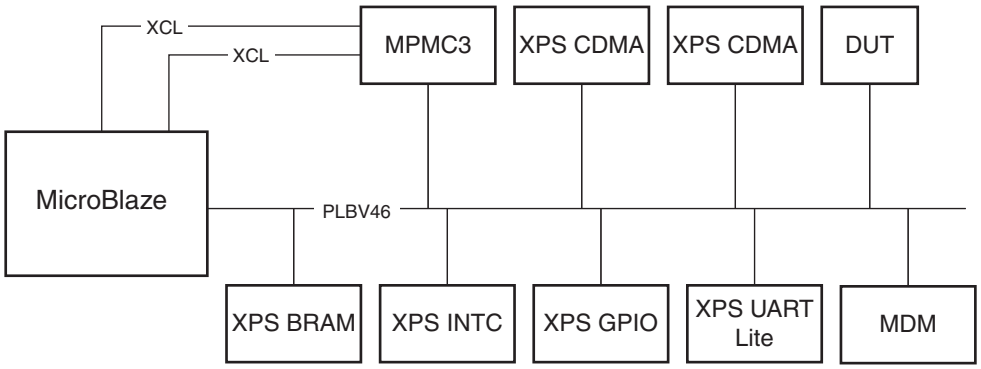

Figure 4: **Virtex-5 LX FPGA System**

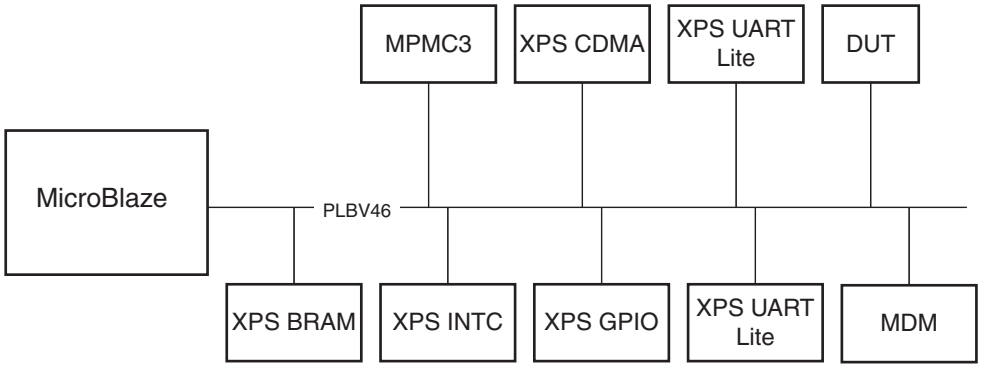

Figure 5: **Spartan-3A DSP/Spartan-6 FPGA System**

The target FPGA was then 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 FMax numbers are shown in Table 30.

Table 30: **PLBv46 PCI Bridge Core System Performance**

Target FPGA	Target f_{MAX} (MHz)
S3A700 -4	90
V4FX60 -10	100
V5LXT50 -1	120

The target fMAX is influenced by the exact system and is provided for guidance. It is not a guaranteed value across all systems.

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. Contact your local Xilinx sales representative for information on pricing and availability of Xilinx LogiCORE IP.

For more information, visit the PLBv46 to PCI Full Bridge 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, 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. Contact your local Xilinx sales representative for information on pricing and availability of Xilinx LogiCORE IP.

Reference Documents

The following documents contain reference information important to understanding the PLBV46 PCI Bridge design. Go to the Xilinx Documentation page to search for Xilinx documents.

- *Xilinx LogiCORE PCI32 Interface v3 and v4 Product Specification*
- *Xilinx LogiCORE PCI v3.0 User Guide*
- *LogiCORE IP v3 PCI32 core Implementation Guide*
- *Xilinx LogiCORE PCI v4.1 User Guide*
- *IBM 128-Bit Processor Local Bus Architecture Specification v4.6*
- *PCI Specification, 2.2 available from the PCI Special Interest Group*
- *PCI32 Design Guide*
- *Virtex-4 FPGA User Guide*
- *Virtex-4 Library Guide*
- *Processor IP Reference Guide*
- LogiCORE IP PCI documents available at Xilinx Solutions for PCI

List of Acronyms

Table 31: **List of Acronyms**

Acronym	Description
BAR	Base Address Register
BE	Byte Enable
DCM	Digital Clock Manager
DISR	Device Interrupt Status Register
DSP	Digital Signal Processing
DUT	Device Under Test
EDIF	Electronic Design Interchange Format
EDK	Embedded Development Kit
FF	Flip-Flop
FIFO	First In First Out
FPGA	Field Programmable Gate Array
GIE	Global Interrupt Enable Register
GPIO	General Purpose Input/Output
I/O	Input/Output
IDSEL	Initialization Device Select
IER	Interrupt Enable Register
IID	Interrupt ID
IPIC	IP Interconnect
IPIF	IP Interface
IPIR	IP Interrupt Register
IPR	Interrupt Pending Register
ISR	Interrupt Status Register
LSB	Least Significant Bit
LUT	Lookup Table
MB	MegaByte

Table 31: **List of Acronyms *(Cont'd)***

Acronym	Description
MHz	Mega Hertz
MPLB	Master Processor Local Bus
MRL	Memory Read Line
MWI	Memory Write Invalidate
NGC	Native Generic Circuit
PCI	Peripheral Component Interconnect
PCIBAR	Peripheral Component Interconnect Base Address Register
PERR	Parity Error
PLB	Processor Local Bus
RAM	Random Access Memory
SERR	System Error
SPLB	Slave Processor Local Bus
SRAM	Static RAM
TCL	Tool Command Language
UCF	User Constraints File
XPS	Xilinx Platform Studio (part of the EDK software)
XST	Xilinx Synthesis Technology

Revision History

Date	Version	Revision
12/11/07	1.0	Initial Xilinx Release
1/10/08	1.1	Changed erroneous OPB reference to PLB in Features section
4/11/08	1.2	Added text re: pull-up resistors to PCI Core Requirements, cross clock constraints to Design Constraints
12/10/08	1.3	Updated to core version v1.03.a and 10.1 design tools.
6/22/11	1.4	Updated to core version v1.04.a and 13.2 design tools.

Notice of Disclaimer