OPB PCI Full Bridge (v1.02a)

DS437 January 25, 2006 **Product Specification**

Introduction

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

The Xilinx OPB is a 32-bit bus subset of the IBM OPB described in the *64-Bit On-Chip Peripheral Bus Architecture Specification v2.0*. Details on the Xilinx OPB, the OPB IPIF, including DMA operation, is found in the *Processor IP Reference Guide*. This guide is accessed via EDK help or the Xilinx website at: http://www.xilinx.com/ise/embedded/proc_ip_ref_guide.pdf.

The LogiCORE PCI v3.0 core provides an interface with the PCI bus. Details of the LogiCORE PCI 32 v3.0 core operation is found in the *Xilinx LogiCORE PCI Interface v3.0 Product Specification* and the *Xilinx The Real-PCI Design Guide v3.0*.

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

Exceptions to the support of PCI commands supported by the v3.0 core are outlined in the Features section.

The OPB 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™ Facts		
Core Specifics		
Supported Device Family	QPro™-R Virtex™-II, QPro Virtex-II, Spartan™-II, Spartan-IIE, Spartan-3 (1), Virtex, Virtex-II, Virtex-E, Virtex-II Pro, Virtex-4	
Version of Core	opb_pci	v1.02a
Resources Used		
Virtex-II	Min	Max
I/O (PCI)	47	50
I/O (OPB-related)	153	155
LUTs	415	4155
FFs	445	2105
Block RAMs	0	2 or more
Provided with Core		
Documentation	Product Specification	
Design File Formats	VHDL	
Constraints File	example UCF-file	
Verification	N/A	
Instantiation Template	N/A	
Reference Designs	None	
Design Tool Requirements		
Xilinx Implementation Tools	8.1.1i or later	
Verification	N/A	
Simulation	ModelSim SE/EE 5.6e or later	
Synthesis	XST	
Support		
Support provided by Xilinx, Inc.		

Notes:
1. Available under Early Access condition only.

Features

- Independent OPB and PCI clocks
- 33 MHz, 32-bit PCI bus support
- Includes a master IP module for PCI initiator transactions, which follows the protocol for interfacing with the OPB IPIF master attachment
- Full bridge functionality
  - OPB Master read and write of a remote PCI target (both single and burst)
  - PCI Initiator read and write to a remote OPB slave (both single and multiple).
  - Supports all PCI commands supported by the v3.0 core with the following exceptions:
    - Interrupt acknowledge command
    - Special cycle command
    - I/O read and I/O write commands are supported only for OPB master read and writes of PCI I/O space as designated by its associated memory designator parameter. All memory space on the OPB-side is designated as memory space in the PCI sense; therefore, I/O commands cannot be used to access memory on the OPB-side.
    - Configuration read and writes are supported (including self-configuration transactions) only when upper word address lines are used for IDSEL lines. 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 to execute the configuration transaction per the PCI 2.2 specification.
- PCI Memory Read Lne (MRL) command is supported in which the v3.0 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 v3.0 core is a target. The v3.0 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 six OPB devices. Each device is defined by an independent set of parameters and a unique OPB memory space:
  - Each device has the following parameters: OPB BAR, high (upper) address, prefetchability, big endian to little endian translate and high-order bits for translation from OPB address space to PCI address space. Byte addressing integrity is maintained by default in all transfers. When configured with FIFOs, address offset is performed by high-order bit substitution. High-order bit definition can be defined with parameters or defined dynamically via registers. When configured without FIFOs, no address translation is performed.
- Supports up to three PCI devices (or BARs in PCI context) with unique PCI memory space. The v3.0 core supports up to three PCI BARs:
  - Each device has the following parameters: PCI BAR, length, prefetchability, memory designator, little endian to big endian translate, and offset for mapping PCI address space to OPB address space. Byte addressing integrity is maintained by default in all transfers. When configured with FIFOs, address offset is performed by high-order bit substitution. High-order bit definition is defined with parameters only. When configured without FIFOs, no address translation is performed

- When configured with FIFOs, byte addressing integrity is maintained by default in all transfers and address translation are done using high-order bit substitution.
- When not configured with FIFOs, only byte addressing integrity is possible (no endianness translation option). In addition, addresses are not translated.
- Address translations are performed by high-order bit substitution. The number of bits substituted depends on the address ranges for both OPB-side and PCI-side.
  - Parameterized selection of high-order bits used in OPB-to-PCI translation can be defined by programmable registers for dynamic translation operation or by parameters for reduced resource utilization
- Available Registers:
  - Interrupt and interrupt enable registers at different hierarchal levels (with FIFOs only)
  - Reset
  - Configuration Address Port, Configuration Data Port, and Bus Number/Subordinate Bus Number
  - Inhibit Transfers on Error (with FIFOs only)
  - OPB Master Address Definition (with FIFOs only)
  - OPB Master Read and Write Addresses (with FIFOs only)
  - High-Order Bits for OPB to PCI address translation (with FIFOs only)
  - Bridge Device Number on PCI bus
- OPB-side Interrupts (with FIFOs only) include
  - OPB Master Read SERR and PERR
  - OPB Master Read Target Abort
  - OPB Master Write SERR and PERR
  - OPB Master Write Target Abort
  - OPB Master Abort Write
  - OPB Master Write Retry and Retry Disconnect
  - OPB Master Write Retry Timeout
  - OPB Master Write Range
  - PCI Initiator Read and Write SERR
- Parameterizable with or without FIFOs:
  - Without FIFOs, yields lower resource utilization, but exhibits low data-throughput because it is performing single transfers only, independently of type of transfer requested
  - With FIFOs, includes asynchronous FIFOs with burst transfer support and backup capability for retrying transfers as needed. High data throughput is realized
- 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 OPB PCI Bridge is

available at the top-level of the OPB 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 I/O-buffers
- 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
- Optional internal DMA. Only simple DMA supported (see the DMA and Scatter Gather product specification). Because the IPIF/DMA operation prefetches data, only prefetchable OPB data should be transferred via DMA from OPB to PCI. This is because the IPIF can, in certain conditions, discard data read from an OPB slave if transfers from a remote PCI agent occurs while the DMA is buffering data to transfer over the PCI bus.
- Input signal to provide the means to asynchronous assert INTR_A from a user supplied register (e.g., an OPB GPIO). The signal, 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 OPB_reset must be simultaneously held at *reset* for at least 5 clock periods of the slowest clock.

Evaluation Version

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

Functional Description

The block diagram of the OPB PCI Bridge design is shown in Figure 1 and described in the following sections. As shown, the OPB PCI Bridge is comprised of three main modules. The On-Chip Peripheral Bus Intellectual Property InterFace (OPB IPIF) interfaces to the OPB bus. The Xilinx LogiCORE PCI32 Interface v3.0 core interfaces to the PCI bus. The IPIF/v3.0 Bridge interfaces between the OPB IPIF and the v3.0 core.

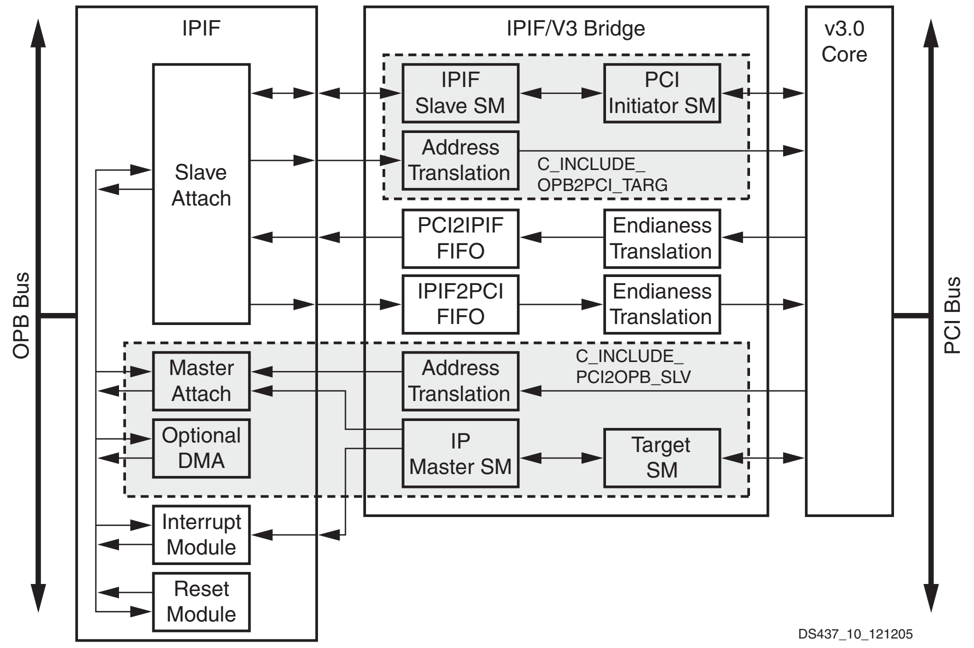

Figure 1: **OPB PCI Full Bridge Block Diagram**

LogiCORE Version 3.0 32-bit PCI Core

The OPB PCI Bridge uses the 32-bit Xilinx LogiCore Version v3.0 IP core. Before the bridge can perform transactions on the PCI bus, the v3.0 core must be configured via configuration transactions from either the PCI-side, or if configuration functionality is included in the bridge configuration, from the OPB-side. Both a Design Guide and an Implementation Guide are available for the Xilinx LogiCore Version v3.0 PCI IP core. These documents detail the v3.0 core operation, including configuration cycles, and are available from Xilinx.

As required by the LogiCore v3.0 core, GNT_N must be asserted for two clock cycles to initiate a PCI transaction by the OPB PCI Bridge.

LogiCore Version 3.0 32-bit PCI Core Requirements

The latency timeout prompt from the v3.0 PCI core is asserted independent of the GNT_N signal state, and due to original v3.0 PCI Core delivery methods, the OPB PCI Bridge was designed to abort initiator transactions based on the v3.0 Core timeout signal. The impact to users is that the OPB PCI Bridge will terminate initiator transactions when the latency timer expires independent of the GNT_N signal state. The bridge does not exploit the option allowed in the PCI Specification that if GNT_N is asserted, the bridge can continue transfers after the latency timer expires. User's have the option to set the latency timer sufficiently large to handle typical burst size to avoid excessive latency timer expirations and repeated requests that would be required to complete a large burst.

Another characteristic of the v3.0 core utilized in the OPB PCI Bridge is that when the OPB PCI bridge needs to initiate an initiator transaction and the arbiter is parked on the OPB PCI Bridge, the v3.0 core will always assert REQ_N independent of GNT_N. The v3.0 core does not exploit the option allowed in the PCI Specification that if GNT_N is asserted and the bus is idle, then the v3.0 core can start another transfer without asserting REQ_N.

Bus Interface Parameters

Because many features in the OPB PCI Bridge design can be parameterized, the user can realize an OPB PCI Full bridge that is uniquely tailored for their system design while using only the resources required for the desired functionality. This approach facilitates achieving the best possible performance with the lowest resource usage. The features that can be parameterized in the OPB PCI Bridge design are shown in Table 1.

Address Translation

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

All address space on the OPB-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 LogiCore PCI v3.0 core supports up to three BARs. Each block has parameters for base address (BAR), PCI length, address translation vector, endianness translation directive, prefetch enable, and memory designator. Only PCI memory space is supported. Both space type and prefetchability must be mirrored in the PCI configuration registers.

Address translations in both directions are performed as high-order address bit substitution for the address vector before crossing to the other bus domain.

- The number of high-order bits substituted in the OPB address that is 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 bits that are used in the substitution from OPB to PCI domains can be defined via the parameter (C_INCLUDE_BAROFFSET_REG) as either a register for each IPIF BAR or a parameter (C_IPIFBAR2PCIBAR_N) for each IPIF BAR.
- The number of high-order bits substituted in the PCI address presented to the bridge for a translation from PCI to OPB domains is given by the bus width minus the parameter C_PCIBAR_LEN_N. The bits that are used in the substitution from PCI to OPB domains are defined by a parameter (C_PCIBAR2IPIFBAR_M) for each BAR.
- The low-order bits are transferred directly between bus domains.

The bridge can be configured via parameters to either maintain byte addressing integrity or to transfer words unaltered from one bus to the other bus. Byte addressing integrity means that if a byte is read from the same memory location by both an OPB master and a PCI initiator, then the same value is obtained. This requires bytes to be swapped with crossing the OPB PCI Bridge. Changing endianess or endianess translation (e.g., little to big endian) is when the word is transferred unaltered from one bus to the other. Both sets of BARs can be independently configured for translation in each direction and each BAR within a set for a given direction is independent. By default, byte addressing integrity is maintained.

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 OPB-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 both IPIF and PCI BAR parameters for bridge configuration *with FIFOs only*.

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

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

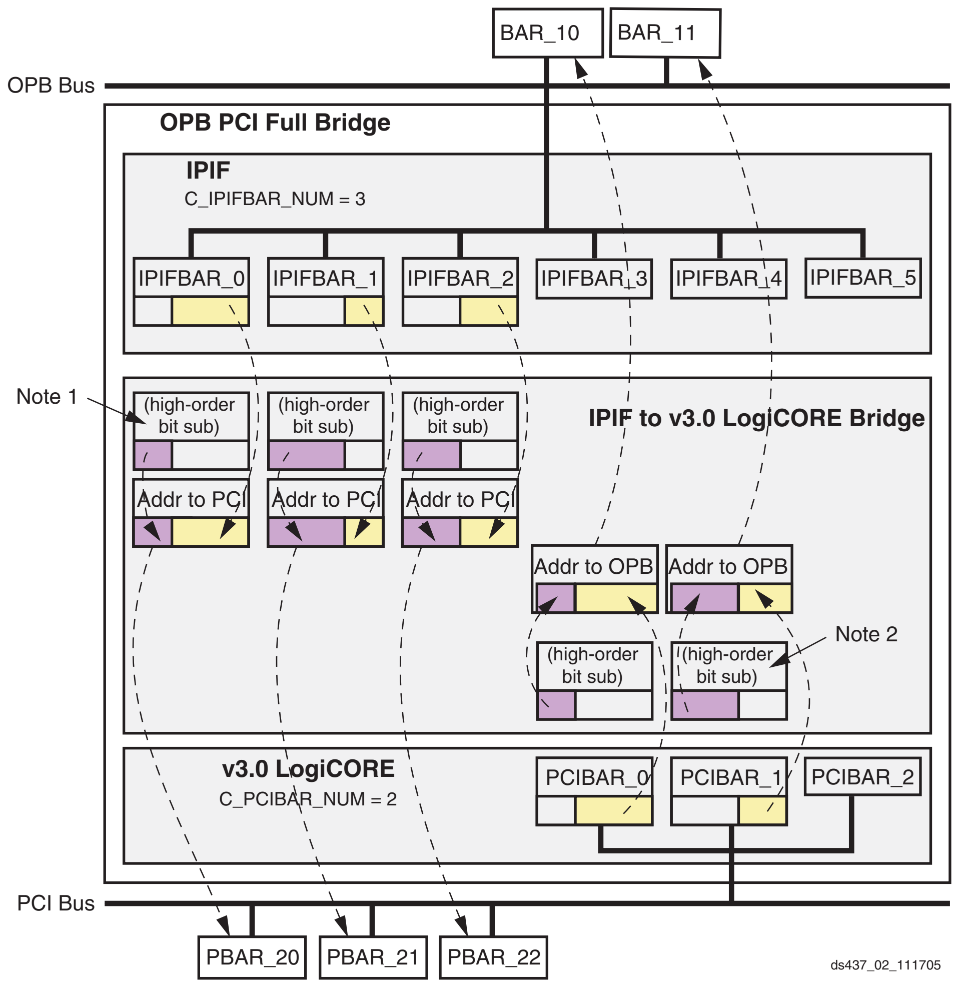

Figure 2: **Translation of Addresses Bus-to-Bus with High-Order Bit Substitution**

Example 1

Because address translations are performed only when the OPB PCI Bridge is configured with FIFOs, the example shown in Figure 2 is for an *OPB 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 OPB bus to the PCI bus.

The OPB parameters are C_IPIFBAR_N, C_IPIF_HIGHADDR_N, C_IPIFBAR2PCIBAR_N, C_IPIFBAR_ENDIAN_TRANSLATE_EN_N, C_IPIF_PREFETCH_N, and C_IPIF_SPACETYPE_N for N=0 to 5.

The PCI parameters are C_PCIBAR_LEN_M, C_PCIBAR2IPIFBAR_M, C_PCIBAR_ENDIAN_TRANSLATE_EN_M, C_IPIF_PREFETCH_M, and C_PCI_SPACETYPE_M for M=0 to M=2.

Example 2 shows of the settings of the two independent sets of base address register (BAR) parameters for specifics of address translation of OPB addresses within the range of a given IPIFBAR to a remote PCI address space. Note that this setting does not depend on the PCIBARs of the OPB 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_IPIF_SPACETYPE_0=1

C_IPIFBAR_1=0xABCDE000
C_IPIF_HIGHADDR_1=0xABCDFFFF
C_IPIFBAR2PCIBAR_1=0xFEDC0xXX (Bits 19-31 are don’t cares)
C_IPIF_SPACETYPE_1=0

C_IPIFBAR_2=0xFE000000
C_IPIF_HIGHADDR_2=0xFFFFFFFF
C_IPIFBAR2PCIBAR_2=0x40xXXXXX (Bits 7-31 are don’t cares)
C_IPIF_SPACETYPE_2=1

C_IPIFBAR_3=0x00000000
C_IPIF_HIGHADDR_3=0x0000007F
C_IPIFBAR2PCIBAR_3=8765438X (Bits 25-31 are don’t cares)
C_IPIF_SPACETYPE_3=1
```

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

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

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

Accessing the OPB PCI Bridge IPIFBAR_3 with address `0x00000071` on the OPB 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 OPB address space. Note that this translation is independent of the OPB PCI Bridge IPIF BARs.

The 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:

```
PCIBAR_0=0xABCDE800 (set by host bridge)
C_PCIBAR_LEN_0=11
C_PCIBAR2IPIFBAR_0=0x123450XX (Bits 21-31 are don't cares)

PCIBAR_1=0x12000000 (set by host bridge)
C_PCIBAR_LEN_1=25
C_PCIBAR2IPIFBAR_1=0xFEXXXXXX (Bits 7-31 are don't cares)
```

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

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

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

Label	Feature / Description	Parameter Name	Allowable Values	Default Value	VHDL Type
Bridge Features Parameter Group (See also G97 to G103)					
G1	Number of IPIF devices	C_IPIFBAR_NUM	1-6; Parameters listed below 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 OPB address (4)	0xFFFFFFFF	std_logic_vector
G3	IPIF BAR high address 0	C_IPIF_HIGHADDR_0	Valid OPB address (4)	0x00000000	std_logic_vector
G4	PCI BAR to which IPIF BAR 0 is mapped unless C_INCLUDE_BAROFFSET_REG=1. If configured without FIFOs, this parameter has no effect.	C_IPIFBAR2PCIBAR_0 (1)	32	0xFFFFFFFF	std_logic_vector
G5	IPIF BAR0 data big endian translation to PCI little endian enable	C_IPIFBAR_ENDIAN_TRANSLATE_EN_0	0=Byte addressing integrity maintained 1=enable translate (allowed only with FIFOs)	0	integer
G6	IPIF BAR 0 prefetchability	C_IPIF_PREFETCH_0	0=not prefetchable 1=prefetchable (only allowed value)	1	integer
G7	IPIF BAR 0 memory/IO designator	C_IPIF_SPACETYPE_0	0=I/O space 1=memory space	1	integer
G8	IPIF device 1 BAR	C_IPIFBAR_1	Valid OPB address (4)	0xFFFFFFFF	std_logic_vector
G9	IPIF BAR high address 1	C_IPIF_HIGHADDR_1	Valid OPB address (4)	0x00000000	std_logic_vector

Table 1: **OPB PCI Bridge Interface Design Parameters *(Contd)***

Label	Feature / Description	Parameter Name	Allowable Values	Default Value	VHDL Type
G10	PCI BAR to which IPIF BAR 1 is mapped unless C_INCLUDE_BAROFFSET_REG=1. If configured without FIFOs, this parameter has no effect.	C_IPIFBAR2 PCIBAR_1	Vector of length C_OPB_AWIDTH (1)	`0xFFFFFFFF`	std_logic_ vector
G11	IPIF BAR1 data big endian translation to PCI little endian enable. See G5 for definition	C_IPIFBAR_ ENDIAN_ TRANSLATE_EN_1	0=Byte addressing integrity maintained 1=enable translate (allowed only with FIFOs)	0	integer
G12	IPIF BAR 1 prefetchability	C_IPIF_ PREFETCH_1	0=not prefetchable 1=prefetchable (only allowed value)	1	integer
G13	IPIF BAR 1 memory and I/O designator	C_IPIF_SPACE TYPE_1	0=I/O space 1=memory space	1	integer
G14	IPIF device 2 BAR	C_IPIFBAR_2	Valid OPB address (4)	`0xFFFFFFFF`	std_logic_ vector
G15	IPIF BAR high address 2	C_IPIF_ HIGHADDR_2	Valid OPB address (4)	`0x00000000`	std_logic_ vector
G16	PCI BAR to which IPIF BAR 2 is mapped unless C_INCLUDE_BAROFFSET_REG=1. If configured without FIFOs, this parameter has no effect.	C_IPIFBAR2 PCIBAR_2	Vector of length C_OPB_AWIDTH (1)	`0xFFFFFFFF`	std_logic_ vector
G17	IPIF BAR2 data big endian translation to PCI little endian enable. See G5 for definition	C_IPIFBAR_ ENDIAN_ TRANSLATE_EN_2	0=Byte addressing integrity maintained 1=enable translate (allowed only with FIFOs)	0	integer
G18	IPIF BAR 2 memory prefetchability	C_IPIF_ PREFETCH_2	0=not prefetchable 1=prefetchable (only allowed value)	1	integer
G19	IPIF BAR 2 memory designator	C_IPIF_SPACE TYPE_2	0=I/O space 1=memory space	1	integer
G20	IPIF device 3 BAR	C_IPIFBAR_3	Valid OPB address (4)	`0xFFFFFFFF`	std_logic_ vector
G21	IPIF BAR high address 3	C_IPIF_ HIGHADDR_3	Valid OPB address (4)	`0x00000000`	std_logic_ vector
G22	PCI BAR to which IPIF BAR 3 is mapped unless C_INCLUDE_BAROFFSET_REG=1. If configured without FIFOs, this parameter has no effect.	C_IPIFBAR2 PCIBAR_3	Vector of length C_OPB_AWIDTH (1)	`0xFFFFFFFF`	std_logic_ vector

Table 1: **OPB PCI Bridge Interface Design Parameters *(Contd)***

Label	Feature / Description	Parameter Name	Allowable Values	Default Value	VHDL Type
G23	IPIF BAR3 data big endian translation to PCI little endian enable. See G5 for definition	C_IPIFBAR_ ENDIAN_ TRANSLATE_EN_3	0=Byte addressing integrity maintained 1=enable translate (allowed only with FIFOs)	0	integer
G24	IPIF BAR 3 memory prefetchability	C_IPIF_ PREFETCH_3	0=not prefetchable 1=prefetchable (only allowed value)	1	integer
G25	IPIF BAR 3 memory designator	C_IPIF_SPACE TYPE_3	0=I/O space 1=memory space	1	integer
G26	IPIF device 4 BAR	C_IPIFBAR_4	Valid OPB address (4)	0xFFFFFFFF	std_logic_ vector
G27	IPIF BAR high address 4	C_IPIF_ HIGHADDR_4	Valid OPB address (4)	0x00000000	std_logic_ vector
G28	PCI BAR to which IPIF BAR 4 is mapped unless C_INCLUDE_BAROFFSET_REG=1. If configured without FIFOs, this parameter has no effect.	C_IPIFBAR2 PCIBAR_4	Vector of length C_OPB_AWIDTH (1)	0xFFFFFFFF	std_logic_ vector
G29	IPIF BAR4 data big endian translation to PCI little endian enable. See G5 for definition	C_IPIFBAR_ ENDIAN_ TRANSLATE_EN_4	0=Byte addressing integrity maintained 1=enable translate (allowed only with FIFOs)	0	integer
G30	IPIF BAR 4 memory prefetchability	C_IPIF_ PREFETCH_4	0=not prefetchable 1=prefetchable (only allowed value)	1	integer
G31	IPIF BAR 4 memory and I/O designator	C_IPIF_SPACE TYPE_4	0=I/O space 1=memory space	1	integer
G32	IPIF device 5 BAR	C_IPIFBAR_5	Valid OPB address (4)	0xFFFFFFFF	std_logic_ vector
G33	IPIF BAR high address 5	C_IPIF_ HIGHADDR_5	Valid OPB address (4)	0x00000000	std_logic_ vector
G34	PCI BAR to which IPIF BAR 5 is mapped unless C_INCLUDE_BAROFFSET_REG=1. If configured without FIFOs, this parameter has no effect.	C_IPIFBAR2 PCIBAR_5	Vector of length C_OPB_AWIDTH (1)	0xFFFFFFFF	std_logic_ vector
G35	IPIF BAR5 data big endian translation to PCI little endian enable. See G5 for definition	C_IPIFBAR_ ENDIAN_TRANSLATE_EN_5	0=Byte addressing integrity maintained 1=enable translate (allowed only with FIFOs)	0	integer
G36	IPIF BAR 5 memory prefetchability	C_IPIF_ PREFETCH_5	0=not prefetchable 1=prefetchable (only allowed value)	1	integer
G37	IPIF BAR 5 memory designator	C_IPIF_SPACE TYPE_5	0=I/O space 1=memory space	1	integer

Table 1: **OPB PCI Bridge Interface Design Parameters *(Contd)***

Label	Feature / Description	Parameter Name	Allowable Values	Default Value	VHDL Type
G38	Number of PCI devices	C_PCIBAR_NUM	1-3; Parameters listed below corresponding to unused BARs are ignored, but must be valid values. BAR label 0 is relevant bar for all values 1-3. G38 must be zero if G98=0.	3	integer
G39	PCI device 0 BAR	C_PCIBAR_0	Don't care (8)		
G40	Power of 2 in the size in bytes of PCI BAR 0 space	C_PCIBAR_LEN_0	5 to 31 (7)	20	integer
G41	IPIF BAR to which PCI BAR 0 is mapped when configured with FIFOs	C_PCIBAR2 IPIFBAR_0	Vector of length C_OPB_AWIDTH (1)	0x000000000	std_logic_ vector
G42	PCI BAR0 data big endian translation to IPIF little endian enable. See G5 for definition	C_PCIBAR_ ENDIAN_ TRANSLATE_EN_0	0=Byte addressing integrity maintained 1=enable translate (allowed only with FIFOs)	0	integer
G43	PCI BAR 0 memory prefetchability	C_PCI_ PREFETCH_0	0=not prefetchable 1=prefetchable (only allowed value)	1	integer
G44	PCI BAR 0 memory designator	C_PCI_SPACE TYPE_0	0=I/O space 1=memory space (only allowed value)	1	integer
G45	PCI device 1 BAR	C_PCIBAR_1	Don't care (8)		
G46	Power of 2 in the size in bytes of PCI BAR 1 space	C_PCIBAR_ LEN_1	5 to 31 (7)	20	integer
G47	IPIF BAR to which PCI BAR 1 is mapped when configured with FIFOs	C_PCIBAR2 IPIFBAR_1	Vector of length C_OPB_AWIDTH (1)	0x000000000	std_logic_ vector
G48	PCI BAR1 data big endian translation to IPIF little endian enable. See G5 for definition	C_PCIBAR_ ENDIAN_ TRANSLATE_EN_1	0=Byte addressing integrity maintained 1=enable translate (allowed only with FIFOs)	0	integer
G49	PCI BAR 1 memory prefetchability	C_PCI_ PREFETCH_1	0=not prefetchable 1=prefetchable (only allowed value)	1	integer
G50	PCI BAR 1 memory and I/O designator	C_PCI_SPACE TYPE_1	0=I/O space 1=memory space (only allowed value)	1	integer
G51	PCI device 2 BAR	C_PCIBAR_2	Don't care (8)		
G52	Power of 2 in the size in bytes of PCI BAR 2 space	C_PCIBAR_LEN_2	5 to 31 (7)	20	integer
G53	IPIF BAR to which PCI BAR 2 is mapped when configured with FIFOs	C_PCIBAR2 IPIFBAR_2	Vector of length C_OPB_AWIDTH (1)	0x000000000	std_logic_ vector

Table 1: **OPB PCI Bridge Interface Design Parameters *(Contd)***

Label	Feature / Description	Parameter Name	Allowable Values	Default Value	VHDL Type
G54	PCI BAR2 data big endian translation to IPIF little endian enable. See G5 for definition	C_PCIBAR_ ENDIAN_ TRANSLATE_EN_2	0=Byte addressing integrity maintained 1=enable translate (allowed only with FIFOs)	0	integer
G55	PCI BAR 2 memory prefetchability	C_PCI_ PREFETCH_2	0=not prefetchable 1=prefetchable (only allowed value)	1	integer
G56	PCI BAR 2 memory designator	C_PCI_SPACE TYPE_2	0=I/O space 1=memory space (only allowed value)	1	integer
G57	PCI address bus width	C_PCI_ABUS _WIDTH	32	32	integer
G58	PCI data bus width	C_PCI_DBUS_ WIDTH	32	32	integer
G59	PCI2IPIF FIFO address bus width	C_PCI2IPIF_ FIFO_ABUS_ WIDTH	0 No FIFOs (2) 5-12 (Virtex) 5-14 (Virtex-II, Virtex-4)	9	integer
G60	IPIF2PCI FIFO address bus width	C_IPIF2PCI_ FIFO_ABUS_ WIDTH	0 No FIFOs (2) 5-12 (Virtex) 5-14 (Virtex -II, Virtex-4))	9	integer
G61	Include INTR_A I/O-buffer	C_INCLUDE_ INTR_A_BUF	0=not included 1=included	1	integer
G62	Include REQ_N I/O-Buffer	C_INCLUDE_ REQ_N_BUF	0=not included 1=included	1	integer
G63	Minimum PCI2IPIF FIFO occupancy level that triggers the bridge to initiate a prefetch PCI read of a remote PCI agent	C_TRIG_PCI_ READ_OCC_ LEVEL	5 to the lesser of 24 or the PCI2IPIF FIFO DEPTH-3 unless DMA is included. Minimum is 16 when DMA is included. PCI2IPIF FIFO DEPTH given by 2^C_PCI2IPIF_FIFO_ABUS_ WIDTH (3)	32	integer
G64	PCI2IPIF FIFO occupancy level that triggers the bridge to initiate an IPIF burst write to remote OPB device	C_TRIG_IPIF_ WRBURST_ OCC_LEVEL	8 (3)	8	integer
G65	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 unless DMA is included. Must be set to 16 when DMA is included. IPIF2PCI FIFO DEPTH given by 2^C_IPIF2PCI_FIFO_ABUS_ WIDTH (3)	16	integer

Table 1: **OPB PCI Bridge Interface Design Parameters *(Contd)***

Label	Feature / Description	Parameter Name	Allowable Values	Default Value	VHDL Type
G66	Minimum IPIF2PCI FIFO vacancy level that inhibits the bridge from initiating a prefetch IPIF burst read of a remote OPB slave	C_INHIBIT_IPIF_READ_VAC_LEVEL	8 (3)	8	integer
G67	Minimum IPIF2PCI FIFO occupancy level that triggers bridge to initiate a prefetch IPIF read of a remote OPB slave	C_TRIG_IPIF_READ_OCC_LEVEL	2 to the lesser of 24 or the IPIF2PCI FIFO DEPTH-3. IPIF2PCI FIFO DEPTH given by 2^C_IPIF2PCI_FIFO_ABUS_WIDTH (3)	16	integer
G68	Number of PCI retry attempts in IPIF posted write operations	C_NUM_PCI_RETRIES_IN_WRITES	Any integer (3)	15	integer
G69	Number of PCI clock periods between retries in posted write operations	C_NUM_PCI_PRDS_BETWN_RETRIES_IN_WRITES	Any integer (3)	15	integer
G70	Obsolete with IPIF update	C_NUM_IPIF_RETRIES_IN_WRITES	Don't care	15	integer
G71	Device base address	C_BASEADDR	Valid OPB address (4), (5)	`0xffffffff`	std_logic_vector
G72	Device absolute high address	C_HIGHADDR	Valid OPB address (4), (5)	`0x00000000`	std_logic_vector
v3.0 core Parameters Group					
G73	PCI Configuration Space Header Device ID	C_DEVICE_ID	16-bit vector	`0x0000`	std_logic_vector
G74	PCI Configuration Space Header Vendor ID	C_VENDOR_ID	16-bit vector	`0x0000`	std_logic_vector
G75	PCI Configuration Space Header Class Code	C_CLASS_CODE	24-bit vector	`0x000000`	std_logic_vector
G76	PCI Configuration Space Header Rev ID	C_REV_ID	8-bit vector	`0x00`	std_logic_vector
G77	PCI Configuration Space Header Subsystem ID	C_SUBSYSTEM_ID	16-bit vector	`0x0000`	std_logic_vector
G78	PCI Configuration Space Header Subsystem Vendor ID	C_SUBSYSTEM_VENDOR_ID	16-bit vector	`0x0000`	std_logic_vector
G79	PCI Configuration Space Header Maximum Latency	C_MAX_LAT	8-bit vector	`0x10`	std_logic_vector
G80	PCI Configuration Space Header Minimum Grant	C_MIN_GNT	8-bit vector	`0x02`	std_logic_vector
DMA Parameters Group					
G81	DMA base address	C_DMA_BASEADDR	Valid OPB address (1), (4), (5), (6)	`0xffffffff`	std_logic_vector

Table 1: **OPB PCI Bridge Interface Design Parameters *(Contd)***

Label	Feature / Description	Parameter Name	Allowable Values	Default Value	VHDL Type
G82	DMA high address	C_DMA_ HIGHADDR	Valid OPB address (1), (4), (5), (6)	0x00000000	std_logic_ vector
G83	Channel type	C_DMA_CHAN_ TYPE	0=simple DMA (1) 9=Do not include DMA	9	integer
G84	Number of bits in the length register	C_DMA_ LENGTH_WIDTH	1 to C_OPB_AWIDTH	11	integer
Configuration					
G85	Include configuration functionality via OPB transactions	C_INCLUDE_ PCI_CONFIG	0=Not included 1=Included	1	integer
G86	Number of IDSEL signals supported	C_NUM_IDSEL	1 to 16	1	integer
G87	PCI address bit connected to PCI v3.0 core IDSEL	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					
G88	Device block ID; intended to be unique for each device in a system	C_DEV_BLK_ ID	0 to 255	6	integer
G89	Allow module identification value to be read from Software Reset module if present.	C_DEV_MIR_ ENABLE	0=disable MIR to save resources 1=enable MIRs	0	integer
G90	Determines whether the interrupt controller includes a Device Interrupt Source Controller	C_INCLUDE_ DEV_ISC	1=include, allow interrupt sources besides the IP (Required with DMA) 0=exclude, allow for only IP interrupts	1	integer
G91	Determines whether the interrupt controller includes an interrupt ID register giving the ordinal value of the highest priority active interrupt	C_INCLUDE_ DEV_PENCODER	1=include 0=Exclude	1	integer
G92	OPB Address width	C_OPB_AWIDTH	32	32	integer
G93	OPB Data width	C_OPB_DWIDTH	32	32	integer
G94	Specifies the target technology	C_FAMILY	Any supported device family	virtex2	string
G95	Include the IPIF interrupt module	C_INCLUDE_ INTR_MODULE	1=Include 0=Exclude	1	integer
G96	Include the IPIF reset module	C_INCLUDE_ RESET_MODULE	1=Include 0=Exclude	1	integer
Bridge Features Parameter Group (See also G1 to G72)					

Table 1: **OPB PCI Bridge Interface Design Parameters *(Contd)***

Label	Feature / Description	Parameter Name	Allowable Values	Default Value	VHDL Type
G97	Include the half-bridge of remote OPB masters accessing PCI targets	C_INCLUDE_ OPB_MST2PCI_ TARG	1=Include 0=Exclude	1	integer
G98	Include the half-bridge of remote PCI initiators accessing OPB slaves	C_INCLUDE_PCI_ INT2OPB_SLV	1=include 0=exclude	1	integer
G99	Include the error recording module when remote OPB master operations are supported (C_INCLUDE_OPB_MS T2PCI_TARG=1)	C_INCLUDE_ ERR_REG_ MODULE	1=include 0=exclude	1	integer
G 100	Include the registers for each IPIFBAR high-order bits to be substituted in translation when remote OPB master operations are supported (C_INCLUDE_OPB_MS T2PCI_TARG=1)	C_INCLUDE_ BAROFFSET_REG	1=include 0=exclude	0	integer
G 101	Include the register for local bridge device number when configuration functionality (C_INCLUDE_PCI_CO NFIG =1) is included	C_INCLUDE_ DEVNUM_REG	1=include 0=exclude	0	integer
G102	Number of PCI clock periods that all remote OPB master operations are retried after a PCI retry occurred in response to a remote PCI initiator write operation. This frees up the IPIF for the specified number of periods to allow the *retried* remote PCI initiator write operation to complete. Only used with FIFOs present.	C_NUMPRD_RETR Y_AFTER_ PCIWRRETRY	Any integer	25	integer
G103	Enables v3.0 core for 66 MHz operation. See v3.0 core manuals for details.	C_66MHZ_ ENABLE	1=66 MHz enabled 0=33 MHz enabled (only allowed value)	0	integer
G104	Number of IDELAY controllers instantiated. Ignored if not a Virtex-4 device	C_NUM_ IDELAYCTRL	2-6 (Virtex-4 only)	2	integer

Table 1: **OPB PCI Bridge Interface Design Parameters *(Contd)***

Label	Feature / Description	Parameter Name	Allowable Values	Default Value	VHDL Type
G105	Includes IDELAY primitive on GNT_N. Set by tcl-scripts and ignored if not Virtex-4 devices.	C_INCLUDE_ GNT_DELAY	1=Include IDELAY primitive (Virtex-4 only) 0=No IDELAY primitive	0	integer
G106	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 effect on bridge functionality.	C_IDELAYCTRL_ LOC	See Device Implementation section, subsection Virtex-4 Support for allowed values	NOT_SET	string

Notes:
1. Not supported when the bridge is configured without FIFOs
2. Specifying either or both IPIF2PCI_FIFO_ABUS_WIDTH=0 or C_PCI2IPIF_FIFO_ABUS_WIDTH=0 configures the bridge without FIFOs.
3. Meaningful only when the bridge is configured with FIFOs.
4. 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.
5. 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.
6. The minimum address range specified by C_DMA_BASEADDR and C_DMA_HIGHADDR must be at least 0x7F.
7. PCI length must follow PCI v2.2 Specification.
8. Carried over from v1.00c to minimize interface change. Has no effect in v1.02a.
9. Endian translate is defined as MSB-to-LSB on OPB(PCI)-side passed unchanged to MSB-to-LSB on PCI(OPB)-side

OPB PCI Bridge Bus Interface I/O Signals

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

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

Port	Signal Name	Interface	I/O	Description
OPB Signals				
P1	OPB_Clk	IPIF	I	OPB clock
P2	OPB_Rst	IPIF	I	OPB reset
P3	IP2INTC_Irpt	IPIF	O	Interrupt output signal
P4	Freeze	IPIF	I	System freeze signal
P5	PCI_DBus(0:C_OPB_ DWIDTH-1)	IPIF	O	Slave output data bus
P6	PCI_xferAck	IPIF	O	Slave transfer acknowledge
P7	PCI_Retry	IPIF	O	Slave retry
P8	PCI_ToutSup	IPIF	O	Slave timeout suppress
P9	PCI_ErrAck	IPIF	O	Slave error acknowledge
P10	OPB_ABus(0:C_OPB_ AWIDTH-1)	IPIF	I	OPB address bus
P11	OPB_BE(0:3)	IPIF	I	OPB byte enables

Table 2: **OPB PCI Bridge I/O Signals *(Contd)***

Port	Signal Name	Interface	I/O	Description
P12	OPB_DBus(0:C_OPB_ DWIDTH-1)	IPIF	I	OPB data bus
P13	OPB_RNW	IPIF	I	Read not Write (logical OR of all master RNW signals)
P14	OPB_select	IPIF	I	Master has taken control of the bus (logical OR of all master selects)
P15	OPB_seqAddr	IPIF	I	OPB sequential address
P16	OPB_errAck	IPIF	I	OPB error acknowledge
P17	OPB_MGrant	IPIF	I	OPB master bus grant
P18	OPB_retry	IPIF	I	OPB retry
P19	OPB_timeout	IPIF	I	OPB timeout error
P20	OPB_xferAck	IPIF	I	OPB transfer acknowledge
P21	PCI_request	IPIF	O	Bridge master bus request
P22	PCI_busLock	IPIF	O	Bridge master bus arbitration lock
P23	PCI_select	IPIF	O	Bridge master select
P24	PCI_RNW	IPIF	O	Bridge master Read not Write
P25	PCI_BE(0:3)	IPIF	O	Bridge master byte enables
P26	PCI_seqAddr	IPIF	O	Bridge master sequential address
P27	PCI_ABus(0:C_OPB_ AWIDTH-1)	IPIF	O	Bridge master address bus
PCI Address and Data Path Signals				
P28	AD[31:0]	PCI Bus	I/O	Time-multiplexed address and data bus
P29	CBE[3:0]	PCI Bus	I/O	Multiplexed bus command and byte enable bus
P30	PAR	PCI Bus	I/O	Generates and checks even parity across AD and CBE
PCI Transaction Control Signals				
P31	FRAME_N	PCI Bus	I/O	Driven by an initiator to indicate a bus transaction
P32	DEVSEL_N	PCI Bus	I/O	Indicates that a target has decoded the address presented during the address phase and is claiming the transaction
P33	TRDY_N	PCI Bus	I/O	Indicates that the target is ready to complete the current data phase
P34	IRDY_N	PCI Bus	I/O	Indicates that the initiator is ready to complete the current data phase
P35	STOP_N	PCI Bus	I/O	Indicates that the target has requested to stop the current transaction
P36	IDSEL	PCI Bus	I	Indicates that the interface is the target of a configuration cycle

Table 2: **OPB PCI Bridge I/O Signals** ***(Contd)***

Port	Signal Name	Interface	I/O	Description
PCI Interrupt Signals				
P37	INTR_A	PCI Bus	O	Indicates that LogiCORE PCI interface requests an interrupt
PCI Error Signals				
P38	PERR_N	PCI Bus	I/O	Indicates that a parity error is detected while the LogiCORE PCI interface is the target of a write transfer or the initiator of a read transfer.
P39	SERR_N	PCI Bus	I/O	Indicates that a parity error was detected during an address cycle, except during special cycles
PCI Arbitration Signals				
P40	REQ_N	PCI Bus	O	Indicates to an external arbiter that the LogiCORE PCI initiator requests access to the bus. Includes I/O-buffer instantiation of standard v3.0 core if C_INCLUDE_REQ_N_BUF=1.
P41	GNT_N	PCI Bus	I	Indicates that the arbiter has granted the bus to the LogiCORE PCI initiator. The input buffer in the standard v3.0 core module is removed.
PCI System Signals				
P42	RST_N	PCI Bus	I	PCI bus reset signal is used to bring PCI-specific registers, sequences, and signals to a consistent state
P43	PCLK	PCI Bus	I	PCI bus clock signal
PCI Bus Internal Arbiter Signals				
P44	INTR_A_int	Internal	O	PCI Bus INTR_A I/O buffer output available at top-level as output from bridge
P45	REQ_N_toArb	Internal	O	Active low PCI Bus REQ_N from OPB PCI Bridge available at top-level as output from bridge. Required for internal PCI arbiter with v3.0 core because REQ_N has an I/O-buffer instantiated in standard v3.0 core.
P46	FRAME_I	Internal	O	PCI Bus FRAME_N I/O buffer output available at top-level as output from bridge
P47	IRDY_I	Internal	O	PCI Bus IRDY_N I/O buffer output available at top-level as output from bridge
User Asserted PCI Interrupt Signal				
P48	Bus2PCI_INTR	Internal		Active high signal to asynchronously assert INTR_A. Inverted signal drives INTR_N user application input of v3.0 core. See v3.0 core documents for details on INTR_N functionality.
Virtex-4 Only, IDELAY Clock				

Table 2: **OPB PCI Bridge I/O Signals** ***(Contd)***

Port	Signal Name	Interface	I/O	Description
P49	RCLK	Internal	I	200 MHz clock input to IDELAY elements of Virtex-4 buffers. Ignored if not Virtex-4 architecture.
PCI Bus Monitoring Debug Vector Signal				
P50	PCI_monitor(0:47)	Internal	O	Output vector to monitor PCI Bus.

The REQ_N_toArb facilitates an interface to an internal PCI arbiter, i.e., in the FPGA. The v3.0 input buffer for GNT_N is removed. This allows an internal connection to GNT_N when using an internal arbiter. When using an external arbiter, GNT_N_fromArb is not needed.

REQ_N is a 3-stated I/O. The REQ_N_toArb port is available to maintain a v3.0 core-like interface. The REQ_N_toArb port allows the use of the same port list for PCI bus interface, implying that the ucf-file for the v3.0 core is the standard file.

The v3.0 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, e.g, an OPB GPIO. If Bus2PCI_INTR is not connected in the mhs-file, then EDK 7.1 tools ties the signal Low. The signal is inverted in the OPB PCI Bridge and AND'ed with the bridge interrupt signal (active Low) to drive the INTR_N input of the v3.0 core. This signal then asynchronously drives INTR_A on the PCI bus. See the v3.0 core specifications on INTR_A behavior relative to v3.0 input INTR_N. The v3.0 core command register *interrupt disable* bit controls the INTR_A operation and v3.0 core status register Interrupt status bit flags if v3.0 core INTR_A is asserted.

Port and Parameter Dependencies

The dependencies between the OPB PCI Bridge design port, i.e., I/O signals, and parameters are shown in Table 3.

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

Label	Parameter	Affects	Depends	Description
Bridge Features Parameter Group (See also G97 to G103)				
G1	C_IPIFBAR_NUM	G8-G37	G97	If G97=0, then G1 must be set to 1 and only G2-G7 are meaningful. If G97=1, then G2-G7 are meaningful and the set of OPB/IPIF BAR-parameters of N=0 to C_IPIFBAR_NUM-1 are meaning. When C_IPIFBAR_NUM < 6, then the parameters of N=C_IPIFBAR_NUM up to and including 5 have no effect. If C_IPIFBAR_NUM=6, then the set of OPB/IPIF BAR-parameters of N=0 to 5 are all meaningful (i.e., G2-G37 are meaningful)
G2	C_IPIFBAR_0	G3	G3	G2 to G3 define range in OPB-memory space that is responded to by this device (IPIF BAR)
G3	C_IPIF_HIGHADDR_0	G2	G2	G2 to G3 define range in OPB-memory space that is responded to by this device (IPIF BAR)
G4	C_IPIFBAR2PCIBAR_0		G2, G3, G59, G60, and G100	Meaningful only if G100=0 and both G59 and G60≠0. In this case, only high-order bits that are the same in G2 and G3 are meaningful

Table 3: **OPB PCI Bridge Parameters-Port Dependencies** ***(Contd)***

Label	Parameter	Affects	Depends	Description
G5	C_IPIFBAR_ENDIAN_ TRANSLATE_EN_0		G59 and G60	Not meaningful if G59 or G60=0.
G6	C_IPIF_PREFETCH_0			Only 1 setting
G7	C_IPIF_SPACETYPE_0			
G8	C_IPIFBAR_1	G9	G1, G9 and G97	Meaningful only if G97=1 and G1>1, then G8 to G9 define the range in OPB-memory space that is responded to by this device (IPIF BAR)
G9	C_IPIF_HIGHADDR_1	G8	G1, G8 and G97	Meaningful only if G97=1 and G1>1, then G8 to G9 define the range in OPB-memory space that is responded to by this device (IPIF BAR)
G10	C_IPIFBAR2PCIBAR_1		G1, G8, G9, G59, G60, G97 and G100	Meaningful only if G1>1, both G59 and G60≠0.G97=1, and G100=0, then only high-order bits that are the same in G8 and G9 are meaningful
G11	C_IPIFBAR_ENDIAN_ TRANSLATE_EN_1		G1, G59 and G60	Meaningful only if G1>1 and both G59 and G60 ≠ 0
G12	C_IPIF_PREFETCH_1		G1 and G97	Meaningful only if G97=1 and G1>1
G13	C_IPIF_SPACETYPE_1		G1 and G97	Meaningful only if G97=1 and G1>1
G14	C_IPIFBAR_2	G15	G1, G15 and G97	Meaningful only if G97=1 and G1>2, then G14 to G15 define the range in OPB-memory space that is responded to by this device (IPIF BAR)
G15	C_IPIF_HIGHADDR_2	G14	G1, G14 and G97	Meaningful only if G97=1 and G1>2, then G14 to G15 define the range in OPB-memory space that is responded to by this device (IPIF BAR)
G16	C_IPIFBAR2PCIBAR_2		G1, G14, G15, G59, G60, G97 and G100	Meaningful only if G1>2, both G59 and G60≠0.G97=1, and G100=0, then only high-order bits that are the same in G14 and G15 are meaningful.
G17	C_IPIFBAR_ ENDIAN_TRANSLATE_EN_2		G1, G59 and G60	Meaningful only if G1>2 and both G59 and G60≠0
G18	C_IPIF_PRE FETCH_2		G1 and G97	Meaningful only if G97=1 and G1>2
G19	C_IPIF_SPACETYPE_2		G1 and G97	Meaningful only if G97=1 and G1>2
G20	C_IPIFBAR_3	G21	G1, G21 and G97	Meaningful only if G97=1 and G1>3, then G20 to G21 define the range in OPB-memory space that is responded to by this device (IPIF BAR)
G21	C_IPIF_HIGHADDR_3	G20	G1, G20 and G97	Meaningful only if G97=1 and G1>3, then G20 to G21 define the range in OPB-memory space that is responded to by this device (IPIF BAR)
G22	C_IPIFBAR2PCIBAR_3		G1, G20, G21, G59, G60, G97 and G100	Meaningful only if G1>3, both G59 and G60≠0.G97=1, and G100=0, then only high-order bits that are the same in G20 and G21 are meaningful
G23	C_IPIFBAR_ENDIAN_ TRANSLATE_EN_3		G1, G59 and G60	Meaningful only if G1>3 and both G59 and G60 ≠ 0

Table 3: **OPB PCI Bridge Parameters-Port Dependencies *(Contd)***

Label	Parameter	Affects	Depends	Description
G24	C_IPIF_PREFETCH_3		G1 and G97	Meaningful only if G97=1 and G1>3
G25	C_IPIF_SPACETYPE_3		G1 and G97	Meaningful only if G97=1 and G1>3
G26	C_IPIFBAR_4	G27	G1, G27 and G97	Meaningful only if G97=1 and G1>4, then G26 to G27 define the range in OPB-memory space that is responded to by this device (IPIF BAR)
G27	C_IPIF_HIGHADDR_4	G26	G1, G26 and G97	Meaningful only if G97=1 and G1>4, then G26 to G27 define the range in OPB-memory space that is responded to by this device (IPIF BAR)
G28	C_IPIFBAR2PCIBAR_4		G1, G26, G27, G59, G60, G97 and G100	Meaningful only if G1>4, both G59 and G60≠0.G97=1, and G100=0, then only high-order bits that are the same in G26 and G27 are meaningful.
G29	C_IPIFBAR_ENDIAN_ TRANSLATE_EN_4		G1, G59 and G60	Meaningful only if G1>4 and both G59 and G60≠0
G30	C_IPIF_PREFETCH_4		G1 and G97	Meaningful only if G97=1 and G1>4
G31	C_IPIF_SPACETYPE_4		G1 and G97	Meaningful only if G97=1 and G1>4
G32	C_IPIFBAR_5	G33	G1, G33 and G97	Meaningful only if G97=1 and G1=6, then G32 to G33 define the range in OPB-memory space that is responded to by this device (IPIF BAR)
G33	C_IPIF_HIGHADDR_5	G32	G1, G32 and G97	Meaningful only if G97=1 and G1=6, then G32 to G33 define the range in OPB-memory space that is responded to by this device (IPIF BAR)
G34	C_IPIFBAR2PCIBAR_5		G1, G32, G33, G59, G60, G97 and G100	Meaningful only if G1=6, both G59 and G60≠0.G97=1, and G100=0, then only high-order bits that are the same in G32 and G33 are meaningful
G35	C_IPIFBAR_ENDIAN_ TRANSLATE_EN_5		G1, G59 and G60	Meaningful only if G1=6 and both G59 and G60≠0
G36	C_IPIF_PREFETCH_5		G1 and G97	Meaningful only if G97=1 and G1=6
G37	C_IPIF_SPACETYPE_5		G1 and G97	Meaningful only if G97=1 and G1=6
G38	C_PCIBAR_NUM	G39-G56	G98	If G98=0, then G38 must be set to 0. If G98=1, then the set of PCI/v3 BAR-parameters of N=0 to C_PCIBAR_NUM-1 are meaningful and the parameters of N=C_PCIBAR_NUM up to and including 2 have no effect. If C_PCIBAR_NUM=3, then the set of PCI/v3 BAR-parameters of N=0 to 2 are all meaningful (i.e., G39-G56 are meaningful)
G39	C_PCIBAR_0			Don't care
G40	C_PCIBAR_LEN_0	G41	G38, G98	Meaningful only if G38>0 and G98=1
G41	C_PCIBAR2IPIFBAR_0		G38, G40, G59, G60, and G98	Meaningful only if G38>0, both G59 and G60≠0 and G98=1, then only the high-order bits above the length defined by G40 are meaningful

Table 3: **OPB PCI Bridge Parameters-Port Dependencies** ***(Contd)***

Label	Parameter	Affects	Depends	Description
G42	C_PCIBAR_ENDIAN_ TRANSLATE_EN_0		G38, G59, G60 and G98	Meaningful only if G38>0, both G59, G60≠0, and G98=1
G43	C_PCI_PREFETCH_0		G38, G59, G60 and G98	Meaningful only if G38>0, both G59 and G60≠0 and G98=1, then only 1 setting
G44	C_PCI_SPACETYPE_0		G38, G59, G60 and G98	Meaningful only if G38>0, both G59 and G60≠0 and G98=1, then only 1 setting
G45	C_PCIBAR_1			Don’t care
G46	C_PCIBAR_LEN_1	G47	G38, G98	Meaningful only if G38>1 and G98=1
G47	C_PCIBAR2IPIFBAR_1		G38, G46, G59, G60 and G98	Meaningful only if G38>1, both G59 and G60≠0 and G98=1, then only the high-order bits above the length defined by G46 are meaningful
G48	C_PCIBAR_ENDIAN_ TRANSLATE_EN_1		G38, G59, G60 and G98	Meaningful only if G38>1, both G59 and G60≠0 and G98=1
G49	C_PCI_PREFETCH_1		G38, G59, G60 and G98	Meaningful only if G38>1, both G59 and G60≠0 and G98=1, then only 1 setting
G50	C_PCI_SPACETYPE_1		G38, G59, G60 and G98	Meaningful only if G38>1, both G59 and G60≠0 and G98=1, then only 1 setting
G51	C_PCIBAR_2			Don’t care
G52	C_PCIBAR_LEN_2	G53	G38, G98	Meaningful only if G38=3 and G98=1
G53	C_PCIBAR2IPIFBAR_2		G38, G52, G59, G60, G98	Meaningful only if G38=3, both G59 and G60≠0 and G98=1, then only the high-order bits above the length defined by G52 are meaningful
G54	C_PCIBAR_ENDIAN_ TRANSLATE_EN_2		G38, G59, G60 and G98	Meaningful only if G38=3, both G59 and G60≠0 and G98=1
G55	C_PCI_PREFETCH_2		G38, G59, G60 and G98	Meaningful only if G38=3, both G59 and G60≠0 and G98=1, then only 1 setting
G56	C_PCI_SPACETYPE_2		G38, G59, G60 and G98	Meaningful only if G38=3, both G59 and G60≠0 and G98=1, then only 1 setting
G57	C_PCI_ABUS_WIDTH			Only 1 setting
G58	C_PCI_DBUS_WIDTH			Only 1 setting

Table 3: **OPB PCI Bridge Parameters-Port Dependencies *(Contd)***

Label	Parameter	Affects	Depends	Description
G59	C_PCI2IPIF_FIFO_ ABUS_WIDTH	G4 G5, G10, G11, G16, G17, G22, G23, G28, G29, G34, G35, G41, G42, G47, G48, G53, G54, G60, G63-G69, G81-G84, G88-G91, G95, and G99-G101	G60 and G94	If G60=0, then G59 defaults to 0. If G59 or G60=0, then GG4 G5, G10, G11, G16, G17, G22, G23, G28, G29, G34, G35, G41, G42, G47, G48, G53, G54, G54, G63-G69, G81-G84, G88-G91, G95, and G99-G101 have no meaning. The maximum is limited by G94.
G60	C_IPIF2PCI_FIFO_ ABUS_WIDTH	G4 G5, G10, G11, G16, G17, G22, G23, G28, G29, G34, G35, G41, G42, G47, G48, G53, G54, G59, G63-G69, G81 -G84, G88-G91, G95, and G99-G101	G59 and G94	If G59=0, then G60 defaults to 0. If G59 or G60=0, then G4 G5, G10, G11, G16, G17, G22, G23, G28, G29, G34, G35, G41, G42, G47, G48, G53, G54, G54, G63-G69, G81-G84, G88-G91, G95, and G99-G101 have no meaning. The maximum is limited by G94.
G61	C_INCLUDE_INTR_A_BUF	P37		If G61=0, then an I/O buffer for P37 is not explicitly instantiated

Table 3: **OPB PCI Bridge Parameters-Port Dependencies** ***(Contd)***

Label	Parameter	Affects	Depends	Description
G62	C_INCLUDE_REQ_N_BUF	P40	G97	If G97=0, then G62 must be set to 0. If G62=0 and G97=1, then an I/O buffer for P40 is not explicitly instantiated
G63	C_TRIG_PCI_READ_ OCC_LEVEL		G59, G60, G83 and G97	If G97=0 or either G59 or G60=0, then G63 has no meaning. If G97=1and both G59 and G60 $\neq 0$, then must be set to 1 to PCI2IPIF FIFO DEPTH-3 unless DMA is included (G83). Minimum is 16 when DMA is included. PCI2IPIF FIFO DEPTH given by 2^{G59}
G64	C_TRIG_IPIF_WRBURST_ OCC_LEVEL		G59, G60 and G98	If G98=0 or either G59 or G60=0, then G64 has no meaning. If G98=1 and both G59 and G60 $\neq 0$, then must be set to 8 which is dictated by IPIF functionality
G65	C_TRIG_PCI_DAT_XFER_ OCC_LEVEL		G59, G60 and G83	If either G59 or G60=0, then G65 has no meaning. If both G59 and G60 $\neq 0$, then must be set to 2 to IPIF2PCI FIFO DEPTH-3 unless DMA is included (G83). Must be set to 16 when DMA is included. IPIF2PCI FIFO DEPTH given by 2^{G60}
G66	C_INHIBIT_IPIF_READ_ VAC_LEVEL		G59, G60 and G98	If G98=0 or either G59 or G60=0, then G66 has no meaning. Meaningful only if G98=1and both G59 and G60 $\neq 0$.
G67	C_TRIG_IPIF_READ_ OCC_LEVEL		G59, G60 and G98	If G98=0 or either G59 or G60=0, then G67 has no meaning. If G97=1 and both G59 and G60 $\neq 0$, then must be set to 1 to IPIF2PCI FIFO DEPTH-3. IPIF2PCI FIFO DEPTH given by 2^{G60}
G68	C_NUM_PCI_RETRIES_ IN_WRITES		G59, G60 and G97	If G97=0 or either G59 or G60=0, then G69 has no meaning
G69	C_NUM_PCI_PRDS_BETWN _RETRIES_IN_WRITES			If G97=0 or either G59 or G60=0, then G68 has no meaning
G70	C_NUM_IPIF_RETRIES_ IN_WRITES			Don't care
G71	C_BASEADDR	G72	G72	G71 to G72 define the range in OPB-memory space that is responded to by OPB PCI Bridge register address space
G72	C_HIGHADDR	G71	G71	G71 to G72 define the range in OPB-memory space that is responded to by OPB PCI Bridge register address space
v3.0 Core Parameters Group				
G73	C_DEVICE_ID			
G74	C_VENDOR_ID			
G75	C_CLASS_CODE			
G76	C_REV_ID			
G77	C_SUBSYSTEM_ID			
G78	C_SUBSYSTEM_ VENDOR_ID			
G79	C_MAX_LAT			
G80	C_MIN_GNT			
DMA Parameters Group				

Table 3: **OPB PCI Bridge Parameters-Port Dependencies** ***(Contd)***

Label	Parameter	Affects	Depends	Description
G81	C_DMA_BASEADDR	G82	G82, G83 and G97	If G59 or G60=0, G83=9 or G97=0, then G81 has no meaning. If G83=0 and G97=1, then G81 to G82 define the range in OPB-memory space that is responded to by OPB PCI integral DMA register address space.
G82	C_DMA_HIGHADDR	G81	G81, G83 and G97	If G59 or G60=0, G83=9 or G97=0, then G81 has no meaning. If G83=0 and G97=1, then G81 to G82 define the range in OPB-memory space that is responded to by OPB PCI integral DMA register address space.
G83	C_DMA_CHAN_TYPE	G81, G82	G97	If G59 or G60=0, then G83 has no meaning. If G97=0, then G83 must be set to 9.
G84	C_DMA_LENGTH_WIDTH			If G83=9 or G97=0, then G81 has no meaning.
Configuration				
G85	C_INCLUDE_PCI_CONFIG	G86, G87 and P36	G97	If G97=0, then G85 has no meaning. If G85=1, then P36 has no internal connection to the port.
G86	C_NUM_IDSEL	G87 and G101	G85, G87 and G97	If G85=0 or G97=0, then G86 has no meaning. If G85=1 and G97=1, then G86 sets the number of devices supported in configuration operations. Must be sufficiently large to include the address bit defined by G87. If G101=1, then G86 restricts the allowed values that are meaningful in the Device Number Register.
G87	C_BRIDGE_IDSEL_ ADDR_BIT	G86	G85, G86, G97 and G101	If G85 or G97=0 or G101=1, then G87 has no meaning. If G85 and G97=1 and G101=0, then G87 must be consistent with the setting of G86.
IPIF Parameters Group				
G88	C_DEV_BLK_ID		G59, G60 and G96	If G59, G60, or G96=0, then G88 has no meaning.
G89	C_DEV_MIR_ENABLE		G59, G60 and G96	If G59, G60, or G96=0, then G89 has no meaning.
G90	C_INCLUDE_DEV_ISC		G59, G60 and G95	If G59, G60, or G95=0, then G90 has no meaning.
G91	C_INCLUDE_DEV_ PENCODER		G59, G60 and G95	If G59, G60, or G95=0, then G91 has no meaning.
G92	C_OPB_AWIDTH			Only 1 setting
G93	C_OPB_DWIDTH			Only 1 setting
G94	C_FAMILY	G59, G60, G104, G105, G106		Limits maximum of G59 and G60. If G94 ≠ Virtex-4, then G104-G106 have no meaning.
G95	C_INCLUDE_INTR_ MODULE		G59, G60 and G97	If G59, G60 or G97=0, then G95 must be set to 0.
G96	C_INCLUDE_ RESET_MODULE		G59 and G60	If G59, G60 or G97=0, then G96 must be set to 0.
Bridge Features Parameter Group (See also G1 to G72)				

Table 3: **OPB PCI Bridge Parameters-Port Dependencies** ***(Contd)***

Label	Parameter	Affects	Depends	Description
G97	C_INCLUDE_OPB_ MST2PCI_TARG	G8-G37, G63, G68, G69, G81-G87, G95, G100, G101, P40, P41		If G97=0, then G8-G37, G63, G68, G69, G81-G87, G100, G101, P40 and P41 have no meaning. If G97=0, then G1 and G98 must be set to 1 and G95 must be set to 0.
G98	C_INCLUDE_PCI_ INT2OPB_SLV	G38-56, G64, G66, G67		If G98=0, then G38-56, G64, G66, G67 have no meaning. If G98=0, then G38 must be set to 0 and G97 must be set to 1.
G99	C_INCLUDE_ERR_ REG_MODULE		G59, G60 and G97	If G59, G60 or G97=0, then G99 has no meaning
G 100	C_INCLUDE_ BAROFFSET_REG	G4, G10, G16, G22, G28 and G34	G59, G60 and G97	If G59, G60, or G97=0, then G100 has no meaning. If G59, G60, and G97 ≠ 0 and G100=1, then G4, G10, G16, G22, G28 and G34 have no meaning. The number of registers is set by G1.
G 101	C_INCLUDE_DEVNUM_REG	G87	G59, G60, G85 and G97	If G59, G60, G85, or G97=0, then G101 has no meaning. If G59, G60, G85, and G97 ≠ 0 and G101=1, then G87 has no meaning. The number of meaningful bits in the Device Number register are defined by G86.
G102	C_NUMPRD_RETRY_ AFTER_PCIWRRETRY			If G98=0, then G102 has no meaning.
G103	C_66MHZ_ENABLE			Only 1 setting
G104	C_NUM_IDELAYCTRL	G106	G94	If G94 ≠ Virtex-4, then G104 has no meaning.
G105	C_INCLUDE_GNT_ DELAY		G94	If G94 ≠ Virtex-4, then G105 has no meaning.
G106	C_IDELAYCTRL_LOC		G94 and G104	If G94 ≠ Virtex-4, then G106 has no meaning. If G94=Virtex-4, then G106 must include the number of LOC coordinates specified by G104.

Supported PCI Bus Commands

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

Table 4: **Supported PCI Bus Commands by LogiCORE and OPB_PCI Bridge**

Command		OPB 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

Table 4: **Supported PCI Bus Commands by LogiCORE and OPB_PCI Bridge *(Contd)***

Command		OPB PCI Bridge	
Code	**Name**	**Target**	**Initiator**
`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

OPB PCI Bridge Registers Descriptions

The OPB 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 calculated by an offset to the base address as shown in Table 5. The registers that exist in a given OPB PCI Bridge implementation depend on the parameter settings of the bridge and is shown in Table 6.

.

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

Register Name	OPB Address	Access
Device Interrupt Status Register (ISR)	C_BASEADDR + `0x00`	Read/TOW
Device Interrupt Pending Register (IPR)	C_BASEADDR + `0x04`	Read/Write
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/TOW
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
Inhibit Transfers on Errors	C_BASEADDR + `0x120`	Read/Write
OPB Mst Error Definition	C_BASEADDR + `0x124`	Read/TOW
OPB Mst Read Address	C_BASEADDR + `0x128`	Read

Table 5: **OPB PCI Bus Interface Registers *(Contd)***

Register Name	OPB Address	Access
OPB Mst Write Address	C_BASEADDR + `0x12C`	Read
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 an OPB PCI Bridge depend on the parameter settings as shown in Table 6.

Table 6: **Register and Parameter Dependencies**

Register Name	Parameter Dependence
Device Interrupt Status Register (ISR)	Present only if G59, G60, G90, G95 and G97 ≠ 0.
Device Interrupt Pending Register (IPR)	Present only if G59, G60, G90, G95 and G97 ≠ 0.
Device Interrupt Enable Register (IER)	Present only if G59, G60, G90, G95 and G97 ≠ 0.
Device Interrupt ID (IID)	Present only if G59, G60, G90, G91, G95 and G97 ≠ 0
Global Interrupt Enable Register (GIE)	Present only if G59, G60, G90, G95 and G97 ≠ 0.
Bridge Interrupt Register	Present only if G59, G60, G95 and G97 ≠ 0.
Bridge Interrupt Enable Register	Present only if G59, G60, G95 and G97 ≠ 0.
Reset Module	Present only if G59, G60 and G96 ≠ 0.
Configuration Address Port	Present only if G85=1.
Configuration Data Port	Present only if G85=1.
Bus Number/Subordinate Bus Number	Present only if G85=1.
Inhibit Transfers on Errors	Present only if G59, G60, G97 and G99 ≠ 0.
OPB Mst Error Definition	Present only if G59, G60, G97 and G99 ≠ 0.
OPB Mst Read Address	Present only if G59, G60, G97 and G99 ≠ 0
OPB Mst Write Address	Present only if G59, G60, G97 and G99 ≠ 0
IPIFBAR2PCIBAR_0 High-Order Bits	Present only if G59, G60, G97 and G100 ≠ 0
IPIFBAR2PCIBAR_1 High-Order Bits	Present only if G1 > 1 and G59, G60, G97 and G100 ≠ 0
IPIFBAR2PCIBAR_2 High-Order Bits	Present only if G1 > 2 and G59, G60, G97 and G100 ≠ 0
IPIFBAR2PCIBAR_3 High-Order Bits	Present only if G1 > 3 and G59, G60, G97 and G100 ≠ 0

Table 6: **Register and Parameter Dependencies *(Contd)***

Register Name	Parameter Dependence
IPIFBAR2PCIBAR_4 High-Order Bits	Present only if G1 > 4 and G59, G60, G97 and G100 ≠ 0
IPIFBAR2PCIBAR_5 High-Order Bits	Present only if G1=6 and G59, G60, G97 and G100 ≠ 0
Host Bridge Device Number	Present only if G59, G60, G85, G97 and G101 ≠ 0

OPB PCI Bridge Interrupt Registers Descriptions

The interrupt module registers are present only if FIFOs are included in the bridge (C_PCI2IPIF_FIFO_ABUS_WIDTH ≠ 0 and C_IPIF2PCI_FIFO_ABUS_WIDTH ≠ 0), as well as the interrupt module (i.e., C_INCLUDE_INTR_MODULE=1 and C_INCLUDE_OPB_MST2PCI_TARG=1). The device interrupt pending register (IPR) and the device interrupt ID register (IID) can be included using C_INCLUDE_DEV_PENCODER and C_INCLUDE_DEV_ISC, respectively. If DMA is included in the bridge, the full interrupt module must be included.

Interrupt Module Specifications

The interrupt registers are in the interrupt module that is instantiated in the IPIF module of the OPB PCI Bridge. Details on the IPIF interrupt module, including discussion of ISR, IPR, IER, and IID, are in the *OPB IPIF Interrupt Product Specification* in the *Processor IP Reference Guide*.

Device Interrupt Source Controller Registers

The device interrupt source controller (ISC) registers (ISR, IPR, IER and IID) must be included if DMA is included.

Global Interrupt Enable Register Descriptions

A global enable is provided to globally enable or disable interrupts from the PCI device. This bit is AND'ed with the output to the interrupt controller. Bit assignment is shown in Table 7. Unlike most other registers, this bit is the MSB on the OPB. This bit is read/write and cleared upon reset.

Table 7: **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	0	**Interrupt Global Enable-** OPB 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	0	Unassigned-

Bridge Interrupt Register Descriptions

The OPB PCI Bridge has thirteen interrupt conditions. Each interrupt can be independently enabled. Bit assignment in the Interrupt register for a 32-bit data bus is shown in Table 8. The interrupt register is read-only and bits are toggled by writing a 1 to the bit(s) being cleared. All bits are cleared upon reset. For more information, see the OPB IPIF Interrupt Product Specification.

Table 8: **Bridge Interrupt Register Bit Definitions (Bit Assignment Assumes 32-Bit Bus)**

Bit(s)	Name	Access	Reset Value	Description
0-18		Read	0	Unassigned
19	PCI Initiator Write SERR	Read/TOW	0	**PCI Initiator Write SERR-** Interrupt(19) indicates that a SERR error was detected during a PCI initiator write of data to an OPB slave.
20	PCI Initiator Read SERR	Read/TOW	0	**PCI Initiator Read SERR-** Interrupt(20) indicates that a SERR error was detected during a PCI initiator read of data from an OPB slave.
21	OPB Master Write FIFO_Overrun	Read/TOW	0	**OPB Master FIFO Overrun-** Interrupt(21) indicates that during a posted OPB Master burst write to a PCI target, the FIFO was overrun. The first address of the transaction when the error occurred is stored. If enabled, subsequent OPB master write operations to a PCI agent are inhibited until error definition register bits are cleared.
22	OPB Master Write Retry Timeout	Read/TOW	0	**OPB Master Burst Write Retry Timeout-** Interrupt(22) indicates that the automatic PCI write retries were not successful due to a latency timeout on the last retry during an OPB Master burst write to a PCI target. The first address of the transaction when the error occurred is stored. If enabled, subsequent OPB master write operations to a PCI agent are inhibited until error definition register bits are cleared.
23	OPB Master Write Retry Disconnect	Read/TOW	0	**OPB Master Burst Write Retry Disconnect-** Interrupt(23) indicates that the automatic PCI write retries were not successful due to a target disconnect on the last retry during an OPB Master burst write to a PCI target. The first address of the transaction when the error occurred is stored. If enabled, subsequent OPB master write operations to a PCI agent are inhibited until error definition register bits are cleared.
24	OPB Master Write Retry	Read/TOW	0	**OPB Master Write Retry-** Interrupt(24) indicates that the automatic PCI write retries were not successful because of a PCI retry on the last retry during an OPB Master burst write to a PCI target. The first address of the transaction when the error occurred is stored. If enabled, subsequent OPB master write operations to a PCI agent inhibited until error definition register bits are cleared.
25	OPB Master Write Master Abort	Read/TOW	0	**OPB Master Write Master Abort-** Interrupt(25) indicates that the OPB PCI Bridge asserted a PCI Master Abort due to no response from a target. The first address of the transaction when the error occurred is stored. If enabled, subsequent OPB master write operations to a PCI agent are inhibited until error definition register bits are cleared.

Table 8: **Bridge Interrupt Register Bit Definitions (Bit Assignment Assumes 32-Bit Bus)** ***(Contd)***

Bit(s)	Name	Access	Reset Value	Description
26	OPB Master Write Target Abort	Read/TOW	0	**OPB Master Write Target Abort-** Interrupt(26) indicates that a PCI Target Abort occurred during an OPB Master Write to a PCI target. The first address of the transaction when the error occurred is stored. If enabled, subsequent OPB master write operations to a PCI agent are inhibited until error definition register bits are cleared.
27	OPB Master Write PERR	Read/TOW	0	**OPB Master Write PERR-** Interrupt(27) indicates that a PERR error is detected on an OPB Master write to a PCI target. The first address of the transaction when the error occurred is stored. If enabled, subsequent OPB master write operations to a PCI agent are inhibited until error definition register bits are cleared.
28	OPB Master Write SERR	Read/TOW	0	**OPB Master Write SERR-** Interrupt(28) indicates that a SERR error was detected by the v3.0 core when performing as a PCI initiator writing data to a PCI target. The first address of the transaction when the error occurred is stored. If enabled, subsequent OPB master write operations to a PCI agent are inhibited until error definition register bits are cleared.
29	OPB Master Read Target Abort	Read/TOW	0	**OPB Master Read Target Abort-** Interrupt(29) indicates that a Target Abort was detected by the v3.0 core when performing as a PCI initiator reading data from a PCI target. The first address of the transaction when the error occurred is stored. If enabled, subsequent OPB master read operations of a PCI agent are inhibited until error definition register bits are cleared.
30	OPB Master Read PERR	Read/TOW	0	**OBP Master Read PERR-** Interrupt(30) indicates that a PERR was detected by the v3.0 core when performing as a PCI initiator reading data from a PCI target. The first address of the transaction when the error occurred is stored. If enabled, subsequent OPB master read operations of a PCI agent are inhibited until error definition register bits are cleared.
31	OPB Master Read SERR	Read/TOW	0	**OPB Master Read SERR-** Interrupt(31) indicates that a SERR error was detected by the v3.0 core when performing as a PCI initiator reading data from a PCI target. The first address of the transaction when the error occurred is stored. If enabled, subsequent OPB master read operations of a PCI agent are inhibited until error definition register bits are cleared.

Bridge Interrupt Enable Register Descriptions

The OPB PCI Bridge interrupt enable features are described in the OPB IPIF Interrupt Product Specification in the Processor IP Reference Guide. Bit assignment in the Bridge Interrupt Enable Register is shown in Table 9. The interrupt enable register is read/write. All bits are cleared upon reset.

Table 9: **Bridge Interrupt Enable Register Bit Definitions (Bit Assignment Assumes 32-Bit Bus)**

Bit(s)	Name	Access	Reset Value	Description
0-18		Read	0	Unassigned-
19	PCI Initiator Write SERR	Read/Write	0	**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	0	**PCI Initiator Read SERR Enable-** Enables this interrupt to be passed to the interrupt controller. • 0 - Not enabled. • 1 - Enabled.
21	OPB Master Write FIFO_Overrun	Read/Write	0	**OPB Master FIFO Overrun-** Enables this interrupt to be passed to the interrupt controller. • 0 - Not enabled. • 1 - Enabled.
22	OPB Master Write Retry Timeout	Read/Write	0	**OPB Master Burst Write Retry Timeout Enable-** Enables this interrupt to be passed to the interrupt controller. • 0 - Not enabled. • 1 - Enabled.
23	OPB Master Write Retry Disconnect	Read/Write	0	**OPB Master Burst Write Retry Disconnect Enable-** Enables this interrupt to be passed to the interrupt controller. • 0 - Not enabled. • 1 - Enabled.
24	OPB Master Write Retry	Read/Write	0	**OPB Master Write Retry Enable-** Enables this interrupt to be passed to the interrupt controller. • 0 - Not enabled. • 1 - Enabled.
25	OPB Master Write Master Abort	Read/Write	0	**Master Write Master Abort Enable-** Enables this interrupt to be passed to the interrupt controller. • 0 - Not enabled. • 1 - Enabled.
26	OPB Master Write Target Abort	Read/Write	0	**OPB Master Write Target Abort Enable-** Enables this interrupt to be passed to the interrupt controller. • 0 - Not enabled. • 1 - Enabled.
27	OPB Master Write PERR	Read/Write	0	**OPB Master Write PERR Enable-** Enables this interrupt to be passed to the interrupt controller. • 0 - Not enabled. • 1 - Enabled.

Table 9: **Bridge Interrupt Enable Register Bit Definitions (Bit Assignment Assumes 32-Bit Bus) *(Contd)***

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

OPB PCI Bridge Reset Descriptions

The IP Reset module is instantiated in the OPB PCI Bridge when the parameter C_INCLUDE_RESET_MODULE is set to 1. C_DEV_MIR_ENABLE controls whether the MIR is included. Details on the IPIF Reset module are in the *Processor IP Reference Guide*. The IP Reset module permits the software reset of the OPB PCI Bridge, independently of other modules in the system, and can include via parameter C_DEV_MIR_ENABLE a register (MIR) for test purposes.

Configuration Address Port Register Description

The Configuration Address Port register exists only if the bridge is configured with PCI host bridge configuration functionality (i.e., C_INCLUDE_PCI_CONFIG=1 and C_INCLUDE_OPB_MST2PCI_TARG=1). This register is read/write, with some bits hardwired, as described in Table 10. Definition of this register is a subset of the PCI 2.2. All accesses to the register are 32-bit accesses. Data is registered 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.

Although only 32-bit read and write OPB operations are supported, byte address integrity is used to define the Configuration Address Port Register bit format. This means that the bytes are swapped as defined in the PCI specification which is little endian word format in defining the OPB-side big endian Configuration Address Port Register bit format

Table 10: **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)

Table 10: **Configuration Address Port Register Bit Definitions (Bit assignment assumes 32-bit bus)**

Bit(s)	Name	Access	Reset Value	Description
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 (i.e., C_INCLUDE_PCI_CONFIG=1 and C_INCLUDE_OPB_MST2PCI_TARG=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 compared with the data in the Bus Number/Subordinate Bus Number register. The fields are defined in Table 11. Reset clears all bits.

As with the Configuration Address Port Register, byte address integrity is used to define the Configuration Data Port Register bit format. This means that the bytes are swapped as defined in the PCI specification which is little endian word format in defining the OPB-side big endian Configuration Data Port Register bit format.

Table 11: **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 (i.e., C_INCLUDE_PCI_CONFIG=1 and C_INCLUDE_OPB_MST2PCI_TARG=1). This register is read/write. All accesses to this 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 register fields are defined in Table 12. Reset clears all bits.

Table 12: **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

Inhibit Transfers on Error Register Description

The Inhibit Transfers on Error Register is present only if FIFOs are included in the bridge (C_PCI2IPIF_FIFO_ABUS_WIDTH ≠ 0 and C_IPIF2PCI_FIFO_ABUS_WIDTH ≠ 0), as well as the error module (i.e., C_INCLUDE_ERR_REG_MODULE=1 and C_INCLUDE_OPB_MST2PCI_TARG=1).

The Inhibit Transfers on Error Register is read/write and allows users to inhibit subsequent transfer if an error is recorded in the OPB Master Error Definition register. Each field of the register which corresponds to a given transaction is a unique bit for enabling the inhibit function.

The two fields correspond to OPB master reads and OPB master writes. Each type of transfer can be inhibited independently. Bit assignment in the Inhibit Transfers on Error Register is shown in Table 13.

Table 13: **Inhibit Transfers on Error Register Bit Definitions (Bit Assignment Assumes 32-Bit Bus)**

Bit(s)	Name	Access	Reset Value	Description
0-29		Read	0	Unassigned
30	Inhibit OPB Master Write Transfers	Read/ Write	0	**Inhibit OPB Master Write Transfers**, when set to 1, inhibits subsequent OPB Master Write transfers when an error has been recorded in the OPB Master Error Address Definition write field. OPB retries are issued when inhibited.
31	Inhibit OPB Master Read Transfers	Read/ Write	0	**Inhibit OPB Master Read Transfers**, when set to 1, inhibits subsequent OPB Master Read transfers when an error has been recorded in the OPB Master Error Address Definition read field. OPB retries are issued when inhibited.

OPB Master Error Definition Register Description

The OPB Master Error Definition register is present if C_INCLUDE_ERR_REG_MODULE=1 and C_INCLUDE_OPB_MST2PCI_TARG=1 and only if FIFOs are included in the bridge (C_PCI2IPIF_FIFO_ABUS_WIDTH ≠ 0 and C_IPIF2PCI_FIFO_ABUS_WIDTH ≠ 0).

The OPB Master Error Definition register contains bit fields that define the meaning of the OPB-side address stored in the OPB Mst Read Address and in the OPB Mst Write Address. This register is read and write 1 to clear. When set, these bits inhibit subsequent OPB Master read or write transfers if *inhibit* is enabled in the Inhibit Transfers on Error Register. The two address registers (read and write) contain addresses for which an error occurred that invoked OPB-side interrupts. Bit assignment in the OPB Master Error Definition register is shown in Table 14.

Table 14: **OPB Master Error Definition Register (Bit Assignment Assumes 32-Bit Bus)**

Bit(s)	Name	Access	Reset Value	Description
0-20		Read	0	Unused.
21	OPB Master Write FIFO_Overrun	Read/TOW	0	**OPB Master FIFO Overrun** indicates that during a posted OPB Master burst write to a PCI target, the FIFO was overrun. The first OPB-side address of the transaction is stored in the OPB Master Write Address register.
22	OPB Master Write Retry Timeout	Read/TOW	0	**OPB Master Write Retry Timeout** indicates that an OPB Master Burst Write Retry Timeout interrupt was asserted during a PCI write operation to the OPB-side address stored in the OPB Master Write Address register.
23	OPB Master Write Retry Disconnect	Read/TOW	0	**OPB Master Write Retry Disconnect** indicates that an OPB Master Burst Write Retry Disconnect interrupt was asserted during a PCI write operation to the OPB-side address stored in the OPB Master Write Address register.
24	OPB Master Write Retry	Read/TOW	0	**OPB Master Write Retry** indicates that an OPB Master Burst Write Retry interrupt was asserted during a PCI write operation to the OPB-side address stored in the OPB Master Write Address register. OPB master write retry is required to meet PCI specification.
25	OPB Master Write Master Abort	Read/TOW	0	**OPB Master Write Master Abort** indicates that an OPB Master Write Master Abort interrupt was asserted during a PCI write operation to the OPB-side address stored in the OPB Master Write Address register.
26	OPB Master Write Target Abort	Read/TOW	0	**OPB Master Write Target Abort** indicates that an OPB Master Write Target Abort interrupt was asserted during a PCI write operation to the OPB-side address stored in the OPB Master Write Address register.
27	OPB Master Write PERR	Read/TOW	0	**OPB Master Write PERR** indicates that an OPB Master Write PERR interrupt was asserted during a PCI write operation to the OPB-side address stored in the OPB Master Write Address register.
28	OPB Master Write SERR	Read/TOW	0	**OPB Master Write SERR** indicates that an OPB Master Write SERR interrupt was asserted during a PCI write operation to the OPB-side address stored in the OPB Master Write Address register.
29	OPB Master Read Target Abort	Read/TOW	0	**OPB Master Read Target Abort** indicates that an OPB Master Read Target Abort interrupt was asserted during a PCI read operation of the OPB-side address stored in the OPB Master Read Address register.
30	OPB Master Read PERR	Read/TOW	0	**OPB Master Read PERR** indicates that an OPB Master Read PERR interrupt was asserted during a PCI read operation of the OPB-side address stored in the OPB Master Read Address register.
31	OPB Master Read SERR	Read/TOW	0	**OPB Master Read SERR** indicates that an OPB Master Read SERR interrupt was asserted during a PCI read operation of the OPB-side address stored in the OPB Master Read Address register.

OPB Master Read Address Register Description

The OPB Master Read Address register is present if C_INCLUDE_ERR_REG_MODULE=1 and C_INCLUDE_OPB_MST2PCI_TARG=1 and only if FIFOs are included in the bridge (C_PCI2IPIF_FIFO_ABUS_WIDTH ≠ 0 and C_IPIF2PCI_FIFO_ABUS_WIDTH ≠ 0).

This register is read-only and is loaded with the OPB-side address that had an error which invoked an interrupt and, if enabled, could inhibit subsequent read operations. The OPB Master Read Address register contains the most recent address experiencing a read error if multiple errors occurred due to inhibiting of subsequent read operations on the first error condition is *not enabled*. Table 15 shows specifics of the data format.

Table 15: **OPB Mst Read Address Register Bit Definitions (Bit assignment assumes 32-bit bus)**

Bit(s)	Name	Access	Reset Value	Description
0-31	D0 - D31	Read-only	0x00	Contains the most recent address experiencing a read error.

OPB Master Write Address Register Description

The OPB Master Write Address register is present if C_INCLUDE_ERR_REG_MODULE=1 and C_INCLUDE_OPB_MST2PCI_TARG =1 and only if FIFOs are included in the bridge (C_PCI2IPIF_FIFO_ABUS_WIDTH ≠ 0 and C_IPIF2PCI_FIFO_ABUS_WIDTH ≠ 0).

This register is read-only and is loaded with the OPB-side address that had an error which invoked an interrupt. The interrupt can inhibit subsequent write operations if the inhibit feature is enabled. If the inhibit feature is not enabled and multiple errors occur, the Master Write Address Register contains the most recent address experiencing a write error. The specifics of the data format are shown in Table 16.

Table 16: **OPB Mst Write Address Register Bit Definitions (Bit Assignment Assumes 32-Bit Bus)**

Bit(s)	Name	Access	Reset Value	Description
0-31	D0 - D31	Read-only	0x00	Contains the most recent address experiencing a write error.

IPIFBAR2PCIBAR_N High-Order Bits Register Description

The IPIFBAR2PCIBAR_N High-Order Bits register is present if C_INCLUDE_BAROFFSET_REG=1 and C_INCLUDE_OPB_MST2PCI_TARG =1 and only if FIFOs are included in the bridge (C_PCI2IPIF_FIFO_ABUS_WIDTH ≠ 0 and C_IPIF2PCI_FIFO_ABUS_WIDTH ≠ 0).

When configured to include the IPIFBAR2PCIBAR_N registers, the values in the registers are used to translate addresses on the OPB bus to the PCI. The register values are used in place 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 that are present is given by the number of IPIF BARs configured (i.e., C_IPIFBAR_NUM). The 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. Stated another way, it is the number of bits that are the same in the C_IPIFBAR_N and C_IPIF_HIGHADDR_N.

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 OPB bus to the PCI bus. Low-order bits pass directly from the OPB 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 OPB 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 OPB PCI Bridge PCI BAR range translation to OPB slaves is not needed because the OPB slave addresses are defined a 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 OPB 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/Writ e	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 OPB 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, there are four registers for the high-order bits to be substituted when making the translation to PCI memory or to I/O 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 seven 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 OPB PCI Bridge IPIFBAR_0 with address 0x12340ABC on the OPB bus yields `0x56710ABC` on the PCI bus.

Writing `0xFEDC0000` to IPIFBAR2PCIBAR_1 High-Order bit register and then accessing the OPB PCI Bridge IPIFBAR_1 with address `0xABCDF123` on the OPB bus yields `0xFEDC1123` on the PCI bus.

Writing `0x40000000` to IPIFBAR2PCIBAR_2 High-Order bit register and then accessing the OPB PCI Bridge IPIFBAR_2 with address `0xFFFEDCBA` on the OPB bus yields `0x41FEDCBA` on the PCI bus.

Writing `0x12345680` to IPIFBAR2PCIBAR_3 High-Order bit register and then accessing the OPB PCI Bridge IPIFBAR_3 with address `0x0000004A` on the OPB bus yields `0x123456CA` on the PCI bus.

Host Bridge Device Number Register Description

The Host Bridge Device Number Register is present only if FIFOs are included in the bridge (C_PCI2IPIF_FIFO_ABUS_WIDTH ≠ 0 and C_IPIF2PCI_FIFO_ABUS_WIDTH ≠ 0), configuration functionality is included (C_INCLUDE_PCI_CONFIG=1 and C_INCLUDE_OPB_MST2PCI_TARG =1), as well as the Host Bridge Device Number Register (i.e., C_INCLUDE_DEVNUM_REG=1).

When configured to include the Host Bridge Device Number Register, the device number of the OPB PCI Bridge is programmed via this register. The register is included in the bridge via the parameter C_INCLUDE_DEVNUM_REG. This register can be included only when C_INCLUDE_PCI_CONFIG=1, which specifies that host bridge functionality is included.

This register is read/write and 4 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 OPB PCI Bridge when configured as a Host Bridge.

OPB PCI Transactions when Configured with FIFOs

The following subsections discuss details of the following types of transactions for a bridge that is configured with FIFOs to realize data throughputs as high as 132 MB/sec. This assumes the OPB clock is 100 MHz or higher. Lower data rates will be realized with lower OPB clock rates for some transactions.

- The section, OPB Master Initiates a Read Request of a PCI target, discusses the OPB master read of a PCI target in which the v3.0 core is the PCI initiator.
- The section, OPB Master Initiates a Write Request to a PCI Target, discusses the OPB master write to a PCI target in which the v3.0 core is the PCI initiator.
- The section, PCI Initiator Initiates a Read Request of an OPB Slave, discusses the remote PCI initiator read of an OPB device in which the v3.0 core is the PCI target
- The section, PCI Initiator Initiates a Write Request to an OPB Slave, discusses the remote PCI initiator write to an OPB device in which the v3.0 core is the PCI target.
- The section, Configuration Transactions, discusses the OPB master read and write of a PCI target configuration space in which the v3.0 core is the PCI initiator.

OPB transactions that are supported are limited to the subset of OPB transactions that are supported by the IPIF. When operating as a master, the IPIF can either perform single transactions (i.e., 1-4 bytes) or bursts of eight words. When the IPIF is operating as a slave, it can perform an arbitrary number of byte, half-word, or word transactions. PCI commands that are supported include I/O read, I/O write, Memory Read, Memory Write, and Memory Read Multiple. Translations of OPB transactions to PCI commands are shown in Table 19. The translations of PCI commands to OPB transactions are shown in Table 20.

OPB timeout suppress is used in remote OPB master read operations due to the latency in the bridge, and potentially slower PCI clock, which would not allow completion of read operations prior to an OPB time-out. Bus lock is used to eliminate arbitration cycles when appropriate.

Table 19: **Translation Table for OPB Transactions to PCI Commands (with FIFOs)**

OPB Master Transaction	PCI I/O Space Prefetchable	PCI Memory Space Prefetchable	PCI Memory Space Non-prefetchable
Single Read	I/O Read	Memory Read	Not supported
Burst Read	I/O Read	Memory Read Multiple	Not supported
Single Write	I/O Write	Memory Write	Not supported
Burst Write	I/O Write	Memory Write	Not supported

Table 20: **Translation Table for PCI Commands to OPB Transactions (with FIFOs)**

PCI Initiator Command	OPB Memory Prefetchable	OPB Memory Non-prefetchable
I/O Read	Not supported	Not supported
I/O Write	Not supported	Not supported
Memory Read	OPB Single Read	Not supported
Memory Read Multiple	OPB Burst Read in groups of eight with all BEs asserted (1)	Not supported
Memory Read Line	OPB Single Read	Not supported

Table 20: **Translation Table for PCI Commands to OPB Transactions (with FIFOs)** ***(Contd)***

PCI Initiator Command	OPB Memory Prefetchable	OPB Memory Non-prefetchable
Memory Write (single data phase)	OPB Single Write	Not Supported
Memory Write (multiple data phase)	OPB Burst Write in groups of eight with < 8 performed as single OPB transactions (2)	Not Supported
Memory Write Invalidate	OPB Single Write	Not Supported

Notes:
1. The IPIF does not support dynamic byte enable (BE) in burst read transactions so when Memory Read Multiple is translated to an OPB burst read, all BEs are asserted during the OPB read operation. Furthermore, the IPIF will only Burst read eight words per OPB transaction.
2. The IPIF will only Burst write eight words per OPB transaction.

The following are system design guidelines in implementing the OPB PCI Bridge in a system.

- PCI and OPB clocks use independent global buffers. For Virtex-4 devices, RCLK must also be driven by global buffer.
- The OPB clock can be slower or faster than the PCI clock. For Virtex-4 devices, RCLK must be 200 MHz.
- Address space on the PCI side that is accessible from the OPB side must be translated to a 2^N contiguous block on the OPB 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, endianness translation directive, prefetch enable, and memory designator (memory or I/O).
- All address space on the OPB 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 LogiCore PCI v3.0 core supports up to 3 BARs. Each block has parameters for base address (BAR), length which must be a 2^N range, address translation vector, endianness translation directive, and prefetch enable. Only memory space in the sense of PCI memory space is supported. Both space type and prefetchability must be 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 system must be designed to accommodate certain throttling restrictions by the OPB PCI Bridge. Both OPB and PCI burst transactions can be broken up into multiple transactions on the target or slave bus because of restrictions on bus protocol and modules in the OPB PCI Bridge. Additional OPB and PCI transactions are automatically initiated when needed to complete a transaction. The first restriction to consider is that the v3.0 core does not permit throttling of data as the initiator or target except for the insertion of wait states prior to the first data transfer. The second restriction is that as a master on the OPB, the OPB PCI Bridge is not allowed to throttle, but the PCI remote initiator can require throttling on the OPB. The OPB PCI Bridge resolves the limitations on throttling by terminating transactions as needed and re-initiating the request to continue as needed. Parameters allow the user to optimize the burst size for high data throughput and to minimize 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. Much of the information in the interrupt status register in the IPIF is duplicated in the OPB PCI Bridge in

the OPB Master Error Definition register. An error condition that results in an interrupt also causes data to be stored in the register. Furthermore, the error condition can result in subsequent bridge function, or functions, to be inhibited if the inhibit bit or bits are set in the Inhibit Transfer on Error register. This register can be used for software control when the bridge operation is inhibited or allowed to continue after an error condition. If Inhibit transfers is enabled, the operation is inhibited until the bit(s) in the OPB Master Error Definition register that were *set* with the interrupt are cleared with a write of a 1 to the bit position corresponding to the operation. The interrupt status register in the IPIF must be cleared with an analogous write.

OPB Master Initiates a Read Request of a PCI target

This section discusses the operation of an OPB master initiating both single reads and burst reads of a remote PCI target. In these transactions, the v3.0 core is the PCI initiator.

The operation is similar whether the PCI space being read is designated as memory or I/O space with the exception of the command sent to the v3.0 core. A parameter associated with each BAR must be consistent with the remote PCI device memory type as either I/O or memory, and based on this parameter setting, either I/O or memory commands are asserted. The OPB master can be the local DMA, a remote OPB master device, or a bridge such as the PLB-to-OPB Bridge. Only one OPB master read of a PCI target is supported at a time.

Commands supported in OPB master read operations are I/O read, memory read, and memory read multiple. The command used is based on the address/qualifier decode, which includes the address, memory type (i.e., I/O or memory type), if OPB_seqAddr is asserted, and the address space prefetchability. Table 19 shows translations of OPB transactions to PCI commands.

The address presented on the OPB 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, i.e., linear incrementing mode. If the PCI target address space is I/O-space, the 2 LSBs are passed unchanged from that presented on the OPB bus.

If the OPB transaction is not a burst, i.e., seqAddr is not High, a single PCI transaction (I/O or memory Read command) is performed. If more data is requested by the OPB master in the same transactions, then another single PCI transaction is initiated. This results in low data throughput.

If the transaction is an OPB burst transaction, i.e., seqAddr is High, and the space type is I/O, then the OPB PCI Bridge issues a IO read command on the PCI bus and a single transaction is performed for all data requested. The data throughput will be low with significant throttling on the OPB bus.

If the transaction is an OPB burst transaction, i.e., seqAddr is High, and the space type is memory, then the OPB PCI Bridge issues a memory read multiple command on the PCI bus and attempts to fill the bridge PCI2IPIF FIFO. Throttling can be performed by the OPB PCI Bridge supplying data to the remote OPB master by delaying acknowledgements until the data is loaded in the FIFO and the data crosses time-domain boundaries to the OPB time-domain. All data is transmitted as soon as it is crossed to the OPB time-domain. Because the PCI bus is usually slower than the OPB, significant throttling time can occur. If the OPB PCI Bridge fills the FIFO in the bridge or the Latency Timer expires, then the OPB PCI Bridge terminates the prefetch read operation. The prefetch read operation can be terminated by the remote PCI target as well.

The user can set the parameter C_TRIG_PCI_READ_OCC_LEVEL to specify when the OPB PCI Bridge starts another prefetch read of the remote PCI target. This parameter specifies the number of words in the FIFO below which the OPB PCI Bridge starts prefetch reads of the remote PCI target. The OPB PCI Bridge ensures that consecutive data is prefetched when new read operations are initiated.

If the PCI2IPIF_FIFO is emptied before more data can be prefetched, an OPB time-out occurs. When the OPB master terminates the transaction with data remaining in the FIFO, the FIFO is flushed. Because the data is required to be prefetchable, data is not lost when the FIFO is flushed.

When an abnormal transaction termination occurs, a bit is set in the OPB Master Error Definition Register. Subsequent OPB master read operations can be inhibited by enabling the inhibit feature for OPB master read operations in the Inhibit Transfers on Error Register.

Upon an abnormal termination, the bit in the OPB Master Error Definition Register that is set must be cleared to enable the OPB read of PCI target operation by writing a 1 to the bit position.

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

Consistent with the PCI specification, OPB masters are required to re-issue commands when a PCI retry is asserted. PCI retries are communicated to the OPB master by asserting OPB_Retry without an interrupt.

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

Abnormal Terminations

When an interrupt is asserted with the error registers included in the bridge configuration, the first address of the transaction when the error occurred is stored the OPB Master Read Address Register and appropriate bits in the OPB Master Error Definition Register are set to identify the error. If enabled, subsequent OPB master read operations of a PCI target are inhibited until the bits in the OPB Master Error Definition Register are cleared by a write of ones to the bits that are set. OPB retries are asserted when the transaction is inhibited.

- If on a single transfer or burst transfer, a parity error occurs during the address phase, then the OPB PCI Bridge causes an OPB timeout for most cases and always asserts the OPB Master Read SERR interrupt. If the parity error on address phase occurs and the target follows the PCI spec recommended response which is to not claim the transactions, then the OPB PCI Bridge terminates the transaction with a Master Abort and an OPB timeout occurs. If the target does not follow PCI specification recommendation and transfers data, then depending on the target decode speed and OPB/PCI clock ratio, data can be transferred with OPB Master Read SERR interrupt being asserted.
- If on a single transfer a SERR occurs during a valid data phase, the OPB PCI Bridge causes an OPB timeout and asserts the OPB Master Read SERR interrupt.
- If on a burst transfers, a SERR occurs during a valid data phase, then the OPB PCI Bridge causes an OPB timeout and asserts the OPB 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. Most of the data transferred prior to the SERR assertion will be transferred. Terminating the data transfer to the OPB master depends on the throttling that is done by the target device and OPB/PCI clock ratio. After the SERR error is transferred across the time-domain boundary, if the *inhibit on error* function is not enabled, an OPB timeout is allowed to occur. If *inhibit on error* is enabled, then an OPB retry is asserted after the FIFO is emptied. In both cases, the OPB Master Read SERR interrupt is asserted.
- If on either a single transfer or burst transfer, the OPB PCI Bridge performs a Master Abort due to no response from a target, an OPB time-out is allowed to occur.

- If on either a single transfer, or the first of a burst transfer, a PCI retry from the PCI target occurs, an OPB retry is issued. Consistent with the PCI specification, the OPB master is required to perform the read again. The OPB PCI Bridge does not retry the read request.
- If the target disconnects with data on a single data transfer, the transfer is completed.
- If a PERR error is detected on a single transfer, data is transferred and the OPB Master Read PERR interrupt is asserted. The PERR status register bit is set as well.
- If the target disconnects on a burst transfer, either with or without data, the v3.0 core terminates the PCI transaction. When the PCI2IPIF FIFO occupancy is below the predetermined level (i.e., C_TRIG_PCI_READ_OCC_LEVEL), another PCI transaction is attempted as long as the OPB master request is active. If a retry is issued on a subsequent PCI transfer, and the OPB master is requesting more data, an automatic retry is issued when the FIFO occupancy is below the predetermined level.
- If a PERR error is detected on a burst transfer, the OPB PCI Bridge aborts the PCI transaction and data transfer to the IPIF is stopped. If the inhibit on error function is not enabled, an OPB timeout is allowed to occur. If inhibit on error is enabled, then an OPB retry is asserted after the FIFO is emptied. In both cases, the OPB Master Read PERR interrupt is asserted. The PERR status register bit is set as well.
- If the initiator Latency Timer expires on a burst transfer, the OPB PCI Bridge terminates the PCI transaction. When the PCI2IPIF FIFO occupancy is below the predetermined level, another PCI transaction is attempted as long as the OPB master request is active.
- If a Target Abort occurs, then data transfer to the IPIF is stopped and OPB timeout suppress is deasserted, which allows an OPB timeout to occur. In addition, the Target Abort OPB Master Read interrupt is asserted. Recall that a Target Abort indicates that the target cannot proceed with subsequent transactions. A Target Abort generally occurs because of a major failure, a situation most likely requiring a reset.
- If the address attempts to go beyond the valid range on a burst transfer, the OPB PCI Bridge terminates the PCI read operation on the last valid address. The FIFO contains only data from valid addresses and transfers to the IPIF continue until the OPB master terminates the transaction or the FIFO is empty. Note that the IPIF will not terminate the burst transaction if the OPB address goes beyond a valid range.

Table 21 summarizes the abnormal conditions when a PCI target can respond and how the response is translated to the OPB master.

Table 21: **Response of OPB Master/v3 Initiator Read of a Remote PCI Target with Abnormal Condition on the PCI Bus**

Abnormal Condition	Single Transfer	Burst (OPB_seqAddr Asserted)
SERR (includes parity error on address phase)	OPB timeout (most cases; see above text) and OPB Master Read SERR interrupt asserted	OPB timeout (most cases; see above text) and OPB Master Read SERR interrupt asserted
OPB PCI Bridge Master abort (no PCI target response)	OPB timeout	OPB timeout
Target Disconnect Without Data (PCI Retry)	OPB Retry. The OPB master must reissue command per PCI specification.	OPB Retry. The OPB master must reissue command per PCI specification.

Table 21: **Response of OPB Master/v3 Initiator Read of a Remote PCI Target with Abnormal Condition on the PCI Bus *(Contd)***

Abnormal Condition	Single Transfer	Burst (OPB_seqAddr Asserted)
Target Disconnect Without Data (after one completed data phase)	N/A	Data is buffered in OPB PCI Bridge PCI2IPIF FIFO. The PCI transaction is terminated by the disconnect. At a predefined FIFO occupancy level, the OPB PCI Bridge issues another PCI transaction at the correct address. If a PCI retry is given, it is automatically retried. The bridge continues to assert PCI_toutSup while trying to get the requested data.
Target Disconnect with Data	Completes	
PERR on data phase	Data is transferred and OPB Master Read PERR interrupt asserted	Data transfer to IPIF is stopped, immediately releasing PCI_toutSup, and asserting OPB Master Read PERR interrupt.
Latency Timer expiration	N/A because v3.0 core waits for one transfer after timeout occurs	Same as target disconnect with/without data
Target Abort	Immediately release PCI_toutSup which allows an OPB timeout and set Target Abort OPB Master Read interrupt	Data transfer to IPIF is stopped, immediately releasing PCI_toutSup, and asserting Target Abort OPB Master Read interrupt.
Address increments beyond valid range	N/A	Stop PCI transaction after last valid address; allow data transfer to IPIF to continue.

OPB Master Initiates a Write Request to a PCI Target

This section discusses the operation of an OPB master initiating both single and burst write transactions to a remote PCI target. Both the single write transactions and burst write transactions are posted writes. In these transactions, the v3.0 core is the PCI initiator.

The write operation is essentially the same whether the PCI space is memory or I/O space; the only difference is the command sent to the v3.0 core by the OPB PCI Bridge. The OPB master can be the local DMA, a remote master device, or a bridge, such as the PLB-to-OPB Bridge. Dynamic byte enable is supported in Xilinx OPB write burst operation and is supported in OPB Master write operations to a PCI target.

Commands supported in OPB 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 (i.e., I/O or memory type) and if OPB_seqAddr is asserted. Table 19 shows translations of OPB transactions to PCI commands.

The address presented on the OPB 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 (i.e., linear incrementing mode). If the PCI target address space is IO-space, the 2 LSBs are passed unchanged from that presented on the OPB bus.

Both single transfers and burst transfers are posted writes 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 v3.0 core, the FIFO usable buffer depth is the actual depth minus 3 words.

Data is loaded in the FIFO on each clock cycle during which the write request is asserted and the address decode is valid. If the transaction is not a burst (i.e., OPB_seqAddr is not High), a single PCI transaction (I/O or memory Write command) is performed. If more data is written by the OPB master, then another single PCI transaction is initiated. In OPB burst transfers (i.e., OPB_seqAddr is asserted), the data is buffered and the PCI transfer is initiated when the FIFO is filled to the level defined by the parameter C_TRIG_PCI_DATA_XFER_OCC_LEVEL or when the OPB write is completed.

In this operation, the v3.0 core is the PCI initiator. The OPB Master cannot throttle data in burst mode, which is a concern during DMA operation, because the data source is a remote OPB slave which can throttle data. If the IPIF2PCI_FIFO empties because the remote OPB slave is throttled, the OPB PCI Bridge must terminate the PCI transaction, then start a new transaction when the IPIF2PCI_FIFO has filled to the parameter-defined occupancy level, or the DMA transaction has completed. To increase data throughput the parameter C_TRIG_PCI_DATA_XFER_OCC_LEVEL must be set to 16 when DMA is included.

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

To inhibit subsequent OPB master write transactions to PCI targets when abnormal terminations occur, set the appropriate bit in the Inhibit Transfers on Error Register. When inhibit is enabled, subsequent transactions are enabled by clearing appropriate bits in the OPB Master Error Definition register.

Consistent with the PCI specification, the OPB PCI Bridge re-issues commands when a PCI retry is asserted. To avoid a permanent *livelock* condition where an infinite series of retries from both directions are asserted, the posted write is attempted to be completed up to a predefined number of retries which is defined by the parameter C_NUM_PCI_RETRIES_IN_WRITES. The re-issue of 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 OPB sources. For example, it must perform single transaction reads of non-prefetchable OPB sources to avoid loss of data in posted writes to a PCI target.

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

Abnormal Terminations

In all cases when an interrupt is asserted with the set of error registers included in the bridge configuration, the first address of the transaction when the error occurred is stored in the OPB Master Write Address Register and appropriate bits in the OPB Master Error Definition Register are set to identify the error. If enabled, subsequent OPB master write operations of a PCI target are inhibited until the bits in the OPB Master Error Definition Register are cleared by a write of ones to the bits that are set. OPB retries are asserted when the transaction is inhibited.

- If a SERR error, including a parity error during the address phase, is detected on either a single transfer or burst transfer, the OPB Master Write SERR interrupt is asserted. If the OPB transfer is in progress, the PCI_errAck is asserted with PCI_xferAck.
- If on either a single write or burst write (i.e., posted OPB to PCI write), the OPB PCI Bridge asserts

a Master Abort because of no response from a target, and the OPB PCI Bridge asserts an OPB Master Write Master Abort interrupt. The IPIF2PCI FIFO is flushed when the Master Abort Write interrupt is asserted.

- If a PCI retry from the PCI target occurs on a single transfer or on the first data cycle of a burst transfer, the OPB PCI Bridge issues a parameterized number of retries (C_NUM_PCI_RETRIES_IN_WRITES). The wait time before a retry is initiated is set by the parameter C_NUM_PCI_PRDS_BETWN_RETRIES_IN_WRITES. During the time that retries are possible, subsequent OPB master write operations to a PCI agent are inhibited and OPB retries are asserted. If the write retry attempts are not successful due to PCI retries, disconnection, or time out, an OPB Master Write Retry interrupt, OPB Master Write Retry Disconnect interrupt, or OPB Master Write Retry Timeout interrupt, respectively, is asserted. The IPIF2PCI FIFO is flushed upon asserting any of the three OPB Master Write interrupts. Consistent with the PCI specification, the OPB master is required to perform the write again if the last of the automatic retries was terminated with a PCI retry.
- If the target disconnects with data on a single transfer, then the transfer completes.
- If the target disconnects, either with or without data after the first data phase, on a burst transfer, then the IPIF/v3.0 core terminates the PCI transaction. If the IPIF2PCI FIFO is not empty, then another PCI transaction is attempted. Due to pipelining in the v3.0 core, the IPIF2PCI_FIFO must backup 1-3 words, depending on the type of target disconnect. The OPB 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 that retries are in progress, subsequent OPB master write operations to a PCI target are inhibited. OPB retries are asserted while the transaction is being retried. If the PCI transaction retries are not successful due to any combination of PCI retries, disconnection, or time out, then an OPB Master Write Retry interrupt, OPB Master Write Retry Disconnect interrupt, or OPB Master Write Retry Timeout interrupt, respectively, will be asserted. The actual interrupt that is asserted 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 OPB Master Write interrupts. Consistent with the PCI specification, the OPB master is required to perform the write again if the last of the automatic retries was terminated with a PCI retry.
- If a PERR error is detected during data phase on a single transfer or on a burst transfer, then the OPB PCI Bridge aborts the PCI transaction and an OPB Master Write PERR interrupt is issued. If the burst transfer is still in progress, PCI_errAck is asserted with PCI_xferAck. The IPIF2PCI FIFO is flushed upon asserting the PERR Write interrupt. The Detected Parity Error status register bit is set as well.
- If the initiator Latency Timer expires on a burst transfer, then the IPIF/v3 terminates the PCI transaction and again attempts to complete the burst write as described above for a PCI retry. A time-out cannot occur during a single transfer because the v3.0 core requires completion of one data transfer after the Latency Timer expires.
- If a Target Abort occurs during either a single write or burst write operation, then the OPB Master Write Target Abort interrupt is asserted. If a burst write is in progress, PCI_errAck is asserted with PCI_xferAck. Target Abort often indicates that the target cannot proceed with subsequent transactions and generally occurs because of a major failure which is a situation most likely requiring a reset.
- If on during a posted OPB to PCI burst write operation the FIFO is filled and data is lost, the OPB

PCI Bridge asserts an OPB Master Write FIFO Overrun interrupt to flag the error.

Table 22 summarizes the abnormal terminations of a PCI target and how the response is translated to the OPB master.

Table 22: **Response of OPB Master/v3 Initiator Write to a Remote PCI Target with Abnormal Condition on PCI Bus**

Abnormal Condition	Single Transfer	Burst (OPB_seqAddr Asserted)
SERR (includes parity error on address phase)	OPB Master Write SERR interrupt asserted[1]	If the transfer is in progress, PCI_errAck is asserted with PCI_xferAck. OPB Master Write SERR interrupt asserted[1] and FIFO flushed.
OPB PCI Bridge Master abort (no PCI target response)	OPB Master Abort Write interrupt asserted[1]and FIFO flushed.	If the transfer is in progress, PCI_errAck is asserted with PCI_xferAck. OPB Master Abort Write interrupt asserted[1] and FIFO flushed.
Target Disconnect Without Data (PCI Retry)	Automatically retries a parameterized number of times. If the last of the PCI write command retries fails due to a PCI retry, the OPB Master Write Retry interrupt is asserted.[1] OPB master must reissue command per PCI specification, if last termination was a retry.	Automatically retries a parameterized number of times. If the last of the PCI write command retries fails due to a PCI retry, the OPB Master Burst Write Retry interrupt is asserted [1] and FIFO flushed. The OPB master must reissue command per PCI specification, when the last termination is a retry.
Target Disconnect Without Data after first 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 OPB Master Burst Write Retry Disconnect interrupt is asserted [1]and FIFO flushed.
Target Disconnect with Data	Completes	
PERR on data phase	Transaction completes and OPB Master Write PERR interrupt asserted[1]	If the transfer is in progress, PCI_errAck is asserted with PCI_xferAck. OPB Master Write PERR interrupt asserted [1] and FIFO is flushed.
Latency Timer expiration	Not applicable because the v3.0 core waits for one transfer after timeout occurs	Automatically retries a parameterized number of times. If the last of the PCI write command retries fails due to a Latency Timer expiration, the OPB Master Burst Write Retry Timeout interrupt is asserted [1] and FIFO is flushed.
Target Abort	Assert OPB Master Write Target Abort interrupt [1] and FIFO is flushed	If the transfer is in progress, PCI_errAck is asserted with PCI_xferAck. OPB Master Write Target Abort interrupt asserted [1] and FIFO is flushed.

Notes:
1. Subsequent OPB master to PCI target write operations, if enabled, are inhibited until the bit associated with the interrupt has been cleared.

PCI Initiator Initiates a Read Request of an OPB Slave

This section discusses the operation of a remote PCI initiator asserting both single read and multiple read commands to read data from a remote OPB slave. In these transactions, the v3.0 core is the PCI target.

Because all OPB 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 to which the OPB PCI Bridge will respond. The I/O read command is ignored, and the configuration read command is responded to by the v3.0 core, but has limited effect on the OPB PCI Bridge.

The OPB PCI Bridge determines if the PCI read command is translated to an OPB read as a burst read or a single read operation based on the PCI command asserted by the PCI initiator. Table 20 shows translations of PCI commands to OPB transactions. During an execution of PCI read commands, an OPB retry from a remote OPB slave is translated to a PCI retry. Only one PCI initiator read of an OPB slave is supported at a time.

For Memory Read commands (i.e., not memory read multiple), the address presented on the PCI is translated to the OPB 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 OPB specification. Byte enables from the PCI bus are passed correctly to the OPB in single OPB read transactions. For Memory Multiple Read and Memory Read Line commands, the address presented on the OPB is word aligned.

Every Memory Read Multiple command that translates to a burst read operation is performed with the full 32 bits on the OPB 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 an OPB slave are ignored, and all byte enables are asserted during the OPB burst read operation. This is an artifact of the IPIF design where all byte enables are asserted on the OPB in burst read operations. Furthermore, dynamic byte enable is not supported by the OPB IPIF for the Xilinx OPB burst read operation and is not supported in PCI initiator burst transactions to OPB slaves. The system designer must ensure that a burst read with all byte enables asserted is not destructive.

PCI initiator read operations from non-prefetchable OPB slaves must be done in a non-destructive way. For example, the user must perform single transaction reads of non-prefetchable OPB slaves to avoid data loss when reading of an OPB slave that is non-prefetchable.

As shown in Table 20, Memory Read commands (i.e., not multiple) are translated to single OPB transactions. A remote PCI initiator can request more than one data transfer with the memory read command, but on the first data phase, the OPB PCI Bridge handles each data request as single OPB transactions with a Disconnect with Data on the PCI bus. This behavior is due to the characteristic of the v3.0 core which does not allow throttling data except as a wait before the first data phase complete. Data throughput will be low when Memory Read commands are utilized.

Data throughput can be very high with Memory Read Multiple transactions. Memory Read Multiple commands of prefetchable memory are translated to repeated OPB burst read transactions of eight words. This behavior is dictated by the IPIF architecture which supports only single word transfers and burst transfers of eight words. Repeated bursts of eight are performed as required in an attempt to fill the IPIF2PCI FIFO until the first word address is seven or less words from the high-address of the remote OPB slave or the PCI initiator ends the transaction. When the high-address is approached, single OPB read requests with all byte enables asserted are performed to read the last seven or less words. After the remote PCI initiator terminates the read transaction, the IPIF2PCI_FIFO is flushed of prefetched data that has not been read by the remote PCI initiator.

When read data is received from a remote OPB slave, the data is loaded in the IPIF2PCI FIFO and synchronized across the OPB/PCI time domain boundary which takes up to two PCI clock cycles to accomplish. The OPB slave can throttle the data read by the remote PCI initiator. If the FIFO is emptied (e.g., the PCI initiator is accepting data faster than the OPB slave is providing it), then the OPB PCI Bridge must Disconnect with Data because the v3.0 core does not allow throttling after the first data phase.

Throttling by the OPB slave and the v3 restriction of not allowing the throttling of data, except as a wait before the first data phase completes, can cause low data throughput. The negative effect on system performance can be minimized by optimizing the parameter that sets the FIFO level when the first data is transferred on the PCI bus during a Memory Read Multiple operation. The parameter is C_TRIG_PCI_XFER_OCC_LEVEL and setting this parameter throttles the first data phase until the FIFO has buffered the number of words set by the parameter. The parameter C_TRIG_PCI_XFER_OCC_LEVEL insures that the transfer is at least this number of words even if the remote OPB slave throttles on the OPB bus.

Another parameter that can increase data throughput is the FIFO occupancy level that triggers the bridge to prefetch more data from the remote OPB slave (C_TRIG_IPIF_READ_OCC_LEVEL). Properly setting this parameter helps insure that the FIFO does not empty while the remote PCI initiator is requesting data.

In a PCI initiator Memory Read Multiple command of an OPB slave, the Master IP module attempts to keep the IPIF2PCI_FIFO full of data read from an OPB slave device for subsequent transfer to the PCI initiator. Data remaining in the FIFO when the PCI initiator terminates the Memory Read Multiple command is discarded. Prefetch is not performed on Memory Read commands (i.e., not Memory Read Multiple).

The OPB PCI Bridge operates the same, independently of whether OPB clock is faster or slower than the PCI clock. Single data request on the PCI bus are translated in the same way to the OPB bus with the only difference being the delays due to the varying clock periods. Because the v3.0 core cannot throttle data flow, the PCI data flow is very different for Memory Read Multiple commands depending on the relative clock speeds. If the OPB clock is faster, the data flow is limited by the PCI bus and the data flow is, in most cases, one continuous Memory Read Multiple transaction.

If the OPB clock is slower, the data flow is a series of PCI transactions that are terminated by the OPB PCI Bridge as a Disconnect Without Data after the number of data phases specified by C_TRIG_PCI_DATA_XFER_OCC_LEVEL, or a few more depending on the OPB slave throttling characteristics and relative clock rates. The series of PCI transactions occurs when the OPB slave does not supply data fast enough for execution of Memory Read Multiple command with single PCI clock cycle data phases. Single clock cycle data phases are required because the v3.0 core cannot throttle the data. The OPB PCI Bridge can throttle the first data transfer to PCI until a predefined number of words are available in the FIFO which is set by C_TRIG_PCI_DATA_XFER_OCC_LEVEL. This parameter will differ for different clock rates and must be adjusted to insure that the PCI specification not be violated. One PCI specification that can be violated is the maximum allowed throttling of the first data transfer.

Abnormal Terminations

- If an address parity error is detected, the v3.0 core will either claim the transaction and issue a Target Abort, or will not claim the transaction and a Master Abort occurs (see v3.0 core documentation). The v3.0 core asserts SERR_N, if enabled, when address phase parity errors are detected.

- 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 might be needed. The OPB PCI Bridge asserts the OPB-side PCI Initiator Read SERR interrupt and continues to supply data if the remote PCI initiator continues to request data.
- If a PERR error is detected on either a single transfer or a burst transfer during a data phase, then the OPB PCI Bridge does nothing. Whether the PCI initiator continues or not depends on the remote PCI initiator behavior.
- If either a Memory Read command or a Memory Read Multiple command is performed and an OPB retry or OPB timeout occurs on the first data phase, the OPB PCI Bridge commands the v3.0 core to Disconnect Without Data or PCI retry, and the PCI initiator is required to retry the transaction. Repeated OPB timeouts, such as addressing a non-existent device, cause a permanent PCI retry response.
- If an OPB retry or OPB timeout occurs on the second or subsequent data phase during a Memory Read Multiple command, then the OPB PCI Bridge automatically retries the OPB request and attempts to keep the FIFO full. If the FIFO is emptied before a retry is successful, the bridge disconnects without data when the FIFO is empty.
- If an OPB ErrAck occurs during either a Memory Read or a Memory Read Multiple command, then the OPB PCI Bridge immediately disconnects without data and the PCI interrupt is strobed.
- On a Memory Read Multiple command transaction, in which the bridge prefectches data, the address will not prefetch beyond the valid range. The OPB PCI Bridge attempts to fill the FIFO with data from addresses up to the limit of the valid range. As data is read by the PCI agent, a Disconnect with Data occurs when the FIFO is emptied. This response is used, rather than a Target Abort, which is an option per the PCI specification. Because the IPIF supports only bursts of 8, single transactions must be done for up to the last seven words (dependent of the first address read). This provides lower data throughput. Depending on relative clock rates, the FIFO can be emptied before the last words are read by single transfers, causing the bridge to disconnect. The v3.0 core cannot throttle as a target after the first data phase. Furthermore, if the first word address is within seven addresses of the upper range limit, the bridge must use IPIF single word transfers. Due to the low data throughput, the bridge is likely to disconnect before all the data is read if the trigger level (C_TRIG_PCI_DATA_XFER_OCC_LEVEL) is lower than seven.

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

Table 23: **Response to PCI initiator Reading a Remote OPB Slave that Terminates the Transfer with an Abnormal Condition on OPB Bus**

Abnormal Condition	Memory Read	Memory Read Multiple
SERR (includes parity error on address phase)	Target abort by the v3.0 core, but completes the OPB transaction. Flush FIFOs and assert OPB-side SERR interrupt.	Target abort by the v3.0 core, but completes the OPB transaction. Flush FIFOs and assert OPB-side SERR interrupt.
PERR	OPB PCI Bridge ignores the signal and continues.	OPB PCI Bridge ignores the signal and continues.
OPB Retry on first data phase	Disconnect Without Data (PCI retry)	Disconnect Without Data (PCI retry).

Table 23: **Response to PCI initiator Reading a Remote OPB Slave that Terminates the Transfer with an Abnormal Condition on OPB Bus *(Contd)***

Abnormal Condition	Memory Read	Memory Read Multiple
OPB Retry after first data phase completes	N/A	Automatically retries OPB read request and attempts to keep FIFO full.
OPB timeout on first data phase	Disconnect Without Data (PCI retry)	Disconnect Without Data (PCI retry).
OPB timeout after first data phase completes	N/A	Automatically retries OPB read request and attempts to keep FIFO full.
OPB ErrAck	Disconnect Without Data, assert PCI interrupt	Disconnect Without Data after any valid buffered data is transferred, assert PCI interrupt.
Address potentially 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 an OPB Slave

This section discusses the operation of a remote PCI initiator asserting the Memory Write command to write data to a remote OPB slave. For these transactions, the v3.0 core is the PCI target.

Because all OPB address space is memory space in the PCI sense, the Memory Write command is the only write command from a remote PCI initiator to which the OPB PCI Bridge responds. The command decode and number words written dictates whether the OPB write operation is a burst or single. Note that byte enables are buffered with data on remote PCI initiator writes to a remote OPB slave. The command I/O write is ignored and the configuration write command is responded to by the v3.0 core, but has limited effect on the OPB 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 reasons for posted writes are that the v3.0 core does not permit data throttling, and that there is not any discriminator for the OPB PCI Bridge to utilize burst OPB command prior to buffering eight words of data. This operation is dictated by the IPIF architecture supporting only single (1) word transfers with arbitrary byte enable bits asserted or bursts of only eight words. Dynamic byte enable is supported in burst writes to remote OPB slaves. To realize maximum data throughput, a multiple of eight words must be transferred on the PCI bus.

Burst writes of eight words to remote OPB slaves are automatically performed by the OPB PCI Bridge. This is accomplished by buffering data in the OPB PCI Bridge until eight words have been accepted. If eight words are buffered, the eight are burst written over the OPB. If the PCI write is terminated before eight words are written, multiple single OPB write transactions are performed to write all the data to the OPB slave device. Large burst write blocks are broken up into multiple combinations of bursts and single OPB transactions as required to transfer all the data.

To realize maximum data throughput, a multiple of eight words must be transferred on the PCI bus. A PCI initiator can write data that is not an integer multiple of eight words, but the OPB PCI Bridge performs the maximum integer number of OPB burst transfers (i.e., eight words per burst), and follows with single OPB transactions to complete the operation. This practice lowers data throughput.

Only one PCI initiator writes to an OPB slave is supported at a time. If the OPB PCI Bridge receives another read or write command from a remote PCI initiator during a write operation, the OPB PCI

Bridge forces a Disconnect Without Data (i.e., PCI retry) until the posted write operation to the remote OPB slave has completed.

If posted writes are pending due to an OPB retry, the IP Master in the bridge attempts to complete the write operation. This implies that the IP Master cannot service PCI read requests while posted writes are being serviced by the automatic write retry feature of the IPIF.

If the write to the remote OPB slave is not successful due to permanent retries by the remote OPB slave, then the IPIF will retry the transaction forever. When this *livelock* condition occurs, the OPB PCI Bridge will retry all other transactions on both the OPB and PCI bus.

Abnormal Terminations

- If an address parity error is detected, the v3.0 core will either claim the transaction and issue a Target Abort, or will not claim the transaction and a Master Abort occurs (see v3.0 core documentation). The v3.0 core asserts SERR_N, if enabled, when address phase parity errors are detected.
- If SERR_N is asserted by a remote agent in a data phase, the bridge disconnects with data on burst writes, and the OPB-side PCI Initiator Write SERR interrupt is asserted. If the SERR occurs after the IP master device has started an OPB transaction, the OPB transaction is immediately terminated. The OPB PCI Bridge flushes any data and resets for a subsequent transaction after terminating the OPB transaction. Any recovery effort that might be needed must be performed on the PCI-side which might require the remote PCI initiator to report the error.
- If a PERR error is detected on write transfer, the v3.0 core asserts PERR signal, if enabled, and sets the Detected PERR error in the status register. The OPB PCI Bridge disconnects with data on the PCI-side during Burst writes. On the OPB-side, the bridge terminates the OPB transfer immediately 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 OPB slave. It is left to the PCI initiator to report the error and initiate any recovery effort that might be needed.
- If an OPB retry occurs while data from the PCI2OPB_FIFO is written to an OPB slave, the IPIF in OPB PCI Bridge attempts to complete writing the buffered data. The wait time between write retries is the OPB arbitration time plus one OPB clock cycle. This is not a parameterized wait time like in the OPB Master to PCI write operation and the number of times the retry is performed in not parameterized. The IPIF will try indefinitely to complete the transaction. Because the IPIF in the OPB PCI Bridge is busy during the retry operation, both PCI initiator reads and writes are inhibited. Target disconnects without data (PCI retry) are asserted for subsequent PCI transactions when the IPIF is attempting to complete the write operation.
- If an OPB timeout or OPB ErrAck occurs while data from the PCI2OPB_FIFO is being written to an OPB slave, the IP Master aborts the OPB transaction, flushes the data, and strobes the PCI interrupt.
- If on a write command transaction the PCI initiator attempts to go beyond the valid address range, the OPB PCI Bridge will not accept data beyond the valid range. Only valid data is buffered in the PCI2OPB_FIFO and all buffered data is transferred to the OPB slave. Because of pipelining in the v3.0 core, Disconnect with Data becomes necessary when the first address nears the end of the valid range, causing the PCI initiator to retry the transfer.

Table 24 summarizes most abnormal conditions with which an OPB slave can respond to a Memory Write command, and how the response is translated to the PCI initiator.

Table 24: **Response to PCI initiator Writing to a Remote OPB Slave that Terminates the Transfer with an Abnormal Condition on OPB Bus**

Abnormal Condition	Memory Write
Parity Error on Address phase	The v3.0 core dictates response with Target Abort or not accepting transaction. SERR_N is asserted if enabled.
SERR on data phase	Disconnect With Data and assert OPB-side PCI Initiator Write SERR interrupt.
PERR on data phase	Single word transfer completes. Burst write transfers are terminated with a Disconnect With Data on the PCI-side and the OPB transfer is terminated.
OPB Retry	Automatically retried an unlimited number of times for each PCI write command.
OPB timeout	Disconnect With Data if PCI transfer is in progress, flush FIFO, and strobe PCI interrupt.
OPB ErrAck	
Address increments beyond valid range	Accept data from only valid address on the PCI bus. Disconnect to terminate transaction.

Configuration Transactions

Functionality for host bridge configuration of PCI agents can be implemented in the OPB PCI Bridge at build time by setting the parameter C_INCLUDE_PCI_CONFIG=1. When the bridge is not configured with host bridge configuration functionality, IDSEL of the v3.0 core is connected to the IDSEL port of the bridge. When the bridge is configured with host bridge configuration functionality, IDSEL of the v3.0 core is connected internally to the specified address signal (as described below) and the IDSEL port of the bridge is not used. As with memory and I/O data transactions, byte addressing integrity (i.e., byte swapping within a word is performed) is maintained in configuration transfers across the bus.

When host bridge configuration functionality is implemented in the OPB PCI Bridge, the v3.0 core in the OPB 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 v3.0 core has the capability to configure only itself until the Bus master enable bit is set in the command register of the v3.0 core and the Latency Timer register is properly set to avoid timeouts. If the v3.0 core Latency Timer is set to 0 value, configuration writes to remote PCI devices will 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`.

Configuration transactions from the OPB-side must be done only when the PCI bus is idle. If a transaction initiated by a remote PCI initiator is being completed (e.g., a posted write to OPB) while an OPB-side configuration operation is initiated, the data can be corrupted. It is the user's resposibility to use configuration operation only after all transactions have completed.

Table 25 shows the results of configuring the v3.0 core configuration header in the OPB PCI Bridge by both OPB-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. Note that OPB-side configuration of the v3.0 core enables all functionality in the Command Status Register and sets the Latency Timer to maximum count for any data value written to the registers. This behavior is an artifact of the v3.0 core behavior.

Configuration Space Header

The LogiCORE v3.0 core used in the OPB 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 below 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 LogiCORE v3 design.

Header Type is a fixed byte of all zeros in the LogiCORE v3 design.

Cardbus CIS Pointer is set to all zeros for the LogiCORE v3 implementation used in the OPB PCI Bridge.

Capabilities Pointer is not enabled for the LogiCORE v3 implementation used in the OPB PCI Bridge.

Interrupt Pin register is set to `0x01`.

BAR3, BAR4 and BAR5 are not supported by the LogiCORE v3.0 core. For these registers and unimplemented PCIBARs (determined by C_PCIBAR_NUM), zeros are returned when read. Writes to the unimplemented configuration space addresses have no effect.

Command/Status, Latency Timer, BAR0, BAR1, and BAR2 registers 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 LogiCORE v3.0 implementation used in the OPB PCI Bridge.

Table 25: **Results of v3.0 Core Command Register Configuration by Remote Host Bridge (PCI-side) and by Self-Configuration (OPB-side)**

	Results in Command Register after Write (OPB-Side Byte Swapped Format)	
Data Written (OPB-side Byte Swapped Format)	**By Remote Host Bridge**	**By Self-Configuration**
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

Notes:

1. This assumes that the PCI BARs in the v3.0 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 v3.0 core to any valid state.

Table 26: **Results of v3.0 Core Latency Timer Register Configuration by Remote Host Bridge (PCI-side) and by Self-configuration (OPB-side)**

	Results in Latency Timer Register after write (OPB-side byte swapped format)	
Data Written	**by remote host bridge**	**by self-configuration**
`0x00`	`0x00`	`0xFF`
`0x01`	`0x01`	`0xFF`
`0xFF`	`0xFF`	`0xFF`

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

The v3.0 core is a PCI 2.2 compliant core, but it has PCI 2.3 compliant features. The v3.0 core documentation should be reviewed for details of compliance.

Configuration transactions from the OPB-side of the bridge are supported by the OPB PCI Bridge. The protocol follows the PCI 2.2 specification but with changes required to adapt to the OPB-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 OPB-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 v3.0 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. An OPB-side read of the Configuration Data Port initiates a Configuration Read command with data returned to the OPB-side upon completion of the PCI-side read command. An OPB-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 nothing occurs on the PCI bus, and a timeout will occur on the OPB bus. This condition is equivalent to the situation where the master enable bit in the configuration command register of the v3.0 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 OPB 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 v3.0 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 the PCI 2.2 specification. Because the v3.0 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 OPB PCI Bridges can be instantiated on a given OPB. Each bridge has a unique base address with fixed offset to a 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 the subordinate buses of the primary PCI bus. Device numbers are independent for each OPB 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 Read/Writes follow closely to single reads/writes by a remote OPB master from/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 OPB master read/writes to remote targets are shown.

Table 27: **Response of OPB Master/v3 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 OPB Master Read SERR interrupt[1]	OPB Master Write SERR interrupt asserted [1]
OPB PCI Bridge Master abort (no PCI target response)	All 1s are returned	OPB Master Abort Write interrupt asserted [1]
Target Disconnect Without Data (PCI Retry)	OPB Retry	Automatically retried a parameterized number of times. If the last of the PCI write command retries fail due to a PCI retry, the OPB Master Burst Write Retry interrupt is asserted.[1] OPB master must reissue command per PCI specification, if last termination was a retry.
Target Disconnect with Data	Completes	Completes
PERR	Data is transferred and OPB Master Read PERR interrupt is asserted	Assert OPB Master Write PERR interrupt asserted [1]
Latency Timer expiration Latency Timer register must be set to non-zero value for accessing remote devices.	N/A because the v3.0 core waits for one transfer after timeout occurs when Latency Timer is non-zero	N/A because the v3.0 core waits for one transfer after timeout occurs when Latency Timer is non-zero
Target Abort	Return all ones and set OPB Master Read Target Abort interrupt [1]	Assert OPB Master Write Target Abort interrupt. [1]

Notes:

1. Subsequent OPB master to PCI target write operations, if enabled, are inhibited until the bit associated with the interrupt has been cleared.

OPB PCI Transactions when Not Configured with FIFOs

The following text discusses details of all types of transactions for a bridge that is configured without FIFOs. In this configuration, the size of the bridge is smaller, and the data throughput is lower. FIFOs are removed by specifying either or both C_IPIF2PCI_FIFO_ABUS_WIDTH=0 or C_PCI2IPIF_FIFO_ABUS_WIDTH=0. DMA cannot be included when the bridge is configured without FIFOs. Stated another way, FIFOs must be included when DMA is included in the bridge IPIF. Host bridge configuration functionality, when included, is identical, independent of whether FIFOs are included or not included.

When the bridge is configured without FIFOs, address translation is not performed. The reason translations are not performed is simply to make the bridge smaller. The address presented on one side of the bridge appears unchanged on the other side with the exception of the two lsbs. The 2 lsbs are set as discussed for the bridge when configured with FIFOs. Another feature that is removed to reduce resource requirement is the option for endianness translation which means that only the byte addressing integrity is available when the bridge is configured without FIFOs. The following parameters have no effect when the bridge is configured without FIFOs:

```
C_IPIFBAR2PCIBAR_N for N= 0 to 5
C_IPIFBAR_ENDIAN_TRANSLATE_EN_N for N=0 to 5
C_PCIBAR2IPIFBAR_M for M=0 to 2
C_PCIBAR_ENDIAN_TRANSLATE_EN_M for M=0 to 2
```

When the bridge is configured without FIFOS, all transactions are single word transactions with byte enables specifying the bytes of interest. Burst transfers on the OPB are handled as single word transfers on the PCI bus. PCI transactions that are intended to be multiple data phase transfers are terminated with a Disconnect with Data on the first data phase. This is an artifact of the v3.0 core behavior which does not allow wait states after the first data phase. Write transfers are not posted (posted write) transfers as with the bridge configured with FIFOS. They require the destination to acknowledge the transfer before completing the transfer on the bus of the master or initiator. Translations of OPB transactions to PCI commands are shown in Table 28. Translations of PCI commands to OPB transactions are shown in Table 29.

Timeout suppress is used in remote OPB master operations due to the latency in the bridge, and possibly slower PCI clock, which would not allow completion of the read operations prior to an OPB time-out. Remote PCI initiator operations do not require asserting time-out suppress.

There are no interrupts when the bridge is configured without FIFOs because all abnormal conditions can be handled in real-time. Abnormal termination on the OPB bus (OPB retry, OPB timeout, OPB ErrAck) result in Disconnect Without Data on the PCI bus. Abnormal terminations on the PCI bus yield essentially the same response, except without the interrupts, as that which is realized when the bridge is configured with FIFOs (see Table 21 and Table 22). To reduce the bridge size, the interrupt module in the IPIF should be removed using the C_INCLUDE_INTR_MODULE parameter.

Table 28: **Translation Table for OPB Transactions to PCI Commands (Without FIFOs)**

OPB Master Transaction	PCI I/O Space Prefetchable	PCI Memory Space Prefetchable
Single Read	I/O Read with Disconnect with Data on first transfer	Memory Read with Disconnect with Data on first transfer
Burst Read	I/O Read with Disconnect with Data on first transfer	Memory Read with Disconnect with Data on first transfer
Single Write	I/O Write with Disconnect with Data on first transfer	Memory Write with Disconnect with Data on first transfer
Burst Write	I/O Write with Disconnect with Data on first transfer	Memory Write with Disconnect with Data on first transfer

Table 29: **Translation Table for PCI Commands to OPB Transactions (Without FIFOs)**

PCI Initiator Command	OPB Memory Prefetchable
I/O Read	Not Supported
I/O Write	Not Supported
Memory Read	OPB Single Read

Table 29: **Translation Table for PCI Commands to OPB Transactions (Without FIFOs)** ***(Contd)***

PCI Initiator Command	OPB Memory Prefetchable
Memory Read Multiple	OPB Single Word Read with Disconnect with Data on first data phase
Memory Read Line	OPB Single Read
Memory Write (single data phase)	OPB Single Write
Memory Write (attempted multiple data phase)	OPB Single Word Write with Disconnect with Data on first transfer
Memory Write Invalidate	OPB Single Write

Design Implementation

Design Tools

The OPB PCI Bridge design is implemented using the VHDL.

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

Design Debug

The OBP PCI Bridge has a test vector output (PCI_monitor) to facilitate system debug (e.g. 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 LogiCORE v3.0 PCI 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, then the wrapper generated by the tools assigns open to this port. PCI Bus monitoring test vector bit definition is listed in Table 30.

Table 30: **PCI Bus Monitoring Signals**

Bit Index	Signal Name	Instantiated IO-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

Table 30: **PCI Bus Monitoring Signals *(Contd)***

Bit Index	Signal Name	Instantiated IO-Buffer
PCI Arbitration Signals		
9	REQ_N	Optional
10	GNT_N	No
PCI Address, Data Path, and Command Signals		
11	PAR	Yes
12-43	AD[31:0]	Yes
44-47	CBE[3:0]	Yes

Design Contraints

The OPB PCI Bridge uses the LogiCORE PCI v3.0 core that requires specific constraints to meet PCI specifications. UCF-files with the constraints for the LogiCORE PCI v3.0 core in many different packages are available from the LogiCORE Lounge. Only pinouts supported by the v3.0 core is supported by the OPB PCI Bridge. If the user is not using the EDK tool flow, then the PCI v3.0 core specific constraints should be included in the top-level ucf-file.

The constraints are implemented automatically in the EDK tool flow with any tool option that invokes bridge synthesis. In this flow, tcl-scripts generate the ucf-file constraints and place them in a file in the OPB PCI Bridge directory of the project implementation directory. The ucf-file constraints are then included in the ngc-file generated in the EDK tool flow. The user can check the ucf-file in the implementation directory of the bridge directory to verify that the constraints are included. As noted above, the user can include all constraints in the top-level ucf-file. When the constraints are included in both the top-level ucf-file and the bridge ngc-file (via the bridge directory ucf-file), then the top-level ucf-file overrides any conflicting constraints in the bridge ngc-file.

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

```
The OPB PCI Bridge design requires design constraints to guarantee performance.
Please refer to the OPB IPIF/LogiCORE PCI64 v3.0 bridge design data sheet for
details.
```

Additional bridge specific constraints are required and an example ucf-file 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-file, PLATGEN will generate 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 LogiCORE PCI v3.0 core require to meet PCI specifications are shown below.

All io buffers must have IOB=TRUE

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

```
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 v3.0 Design Guide and v3.0 core ucf-files. These parameters are automatically set in the normal EDK tool flow, but can be included in the system top-level ucf-file. For alternative tool flows, the settings are shown below. When the complete set of constraints is used, the PCI clock must be a PAD input which is the required clock routing for all v3.0 core implementations. The EDK flow checks if the PCI clock is a PAD input and if it is, then the OFFSET constraints shown below are includes 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 devices: QPro-R, Virtex-II, QPro Virtex-II, Spartan-II, Spartan-IIE, Virtex, Virtex-II, Virtex-E, Virtex-II Pro, and Virtex-4.

Virtex-4 Support

To meet PCI specification setup and hold times with the Virtex-4 architecture, 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-file. When IDELAY primitives are used in the mode required by the LogiCORE v3.0 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. Note that these primitives are only required for Virtex-4 architecture. The additional constraints are discussed after the discussion of primitives specific to Virtex-4 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 platform, the port does not connect to anything in the OPB PCI Bridge, and it can 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 OPB_RST is asserted to the OPB PCI Bridge. An unstable clock can result failure of OPB 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 LogiCORE v3.0 core, as well as reference designs using the OPB PCI Bridge, present options for generating the 200 MHz clock.

IDELAY primitives are instantiated automatically by the bridge when the Virtex-4 C_FAMILY parameter is set to the Virtex-4 architecture. The EDK tools automatically sets the C_FAMILY parameter and it can not 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 (e.g., 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 is accomplished with a tcl-script in the OPB 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 User Guide discussion of IDELAYCTRL usage and design guidance for more details on IDELAYCTRL and usage. Tools beyond ISE 8.1 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 architecture. The actual number depends on the pinout defined by the user. To avoid the undesirable results noted above, the LogiCORE v3 PCI core standalone 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 OPB PCI Bridge allows specifying the number of IDELAYCTRL primitives from two to six; this is set at build time by setting the parameter C_NUM_IDELAYCTRL. However, it might be difficult to meet timing when the pinout is spread out to require three 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 will occur 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-file LOC constraints is shown in the example below where the instance name in the OPB 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 example below is for a design that uses 2 IDELAYCTRL primitives.

This approach allows users to use the constraint LOC coordinates directly from the LogiCORE v3.0 core ucf-generator. Note that the ucf-file generator prescribes I/O pin layout that only uses two IDELAYCTRL primitives. The example below 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 will generate constraints in the bridge core ucf-file that is combined with the opb_pci bridge ngc-file during normal EDK tool flow. Note that if the LOC constraints are set in the system top-level ucf-file, then this parameter is has no effect for either case of it being properly set or set to default (i.e., NOT_SET). This is because the system top-level ucf-file 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 OPB PCI Bridge. If the system top-level ucf-file 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 below.

The parameter C_IDELAYCTRL_LOC has the syntax of IDELAYCTRL_XNYM where N and M are coordinates and multiple entries are concatenated by - (i.e., 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-file for the above example, the parameter should be set in the MHS-file as shown below.

```
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 the 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-file. For alternative tool flows, the setting are shown below. The settings shown below are settings at the time this document was written. The LogiCORE v3 PCI core Implementation Guide and v3.0 core ucf generator tool should be checked for updated values. IOSTANDARD must be explicitly defined in the ucf-file 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 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-file constraints and place them in a file in the OPB PCI Bridge directory of the project implementation directory. The ucf-file constraints are then included in the ngc-file generated in the EDK tool flow. The user can check the ucf-file 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-file. When the constraints are included in both the top-level ucf-file and the bridge ngc-file (via the bridge directory ucf-file), then the top-level ucf-file overrides any conflicting constraints in the bridge ngc-file.

Device Utilization and Performance Benchmarks

Because the OPB PCI Bridge is used with other design pieces in the FPGA, the utilization and timing numbers reported are estimates only. As the OPB PCI Bridge is combined with other pieces of the FPGA design, the utilization of FPGA resources and timing of the OPB PCI Bridge design will vary.

To analyze the OPB PCI Bridge timing within the FPGA, a design outlined in Table 31 instantiated the OPB PCI Bridge with the parameters listed. The data shown if for a Virtex-II Pro device. An additional GCLK is required for the RCLK 200 MHz signal for Virtex-4 devices.

Table 31: ***OPB PCI Bridge*** **FPGA Performance and Resource Utilization Benchmarks**

	Parameter Values						**Device Resources**					**fMAX**
Configuration Description	C_IPIFBAR_NUM	C_PCI_BAR_NUM	C_IPIF2PCI_FIFO_ABUS_WIDTH	C_INCLUDE_PCI_CONFIG	C_INCLUDE_OPB_MST2PCI_TARG	C_INCLUDE_PCI_INT2OPB_SLV	Slices	Slice Flip- Flops	4- input LUTs	# BRAM	# GCLK	MHz
IPIF (no DMA, with Burst)	6	N/A	N/A	N/A	N/A	N/A	473	311	666	0	1 (OPB)	>100
IPIF (with DMA, with Burst)	6	N/A	N/A	N/A	N/A	N/A	1018	694	1571	0	1 (OPB)	>100
OPB PCI Bridge (with FIFOs and Burst support)	6	3	9	1	1	1	1445	858	1954	2	2 (OPB, PCI)	>100
v3.0 core	N/A	3	N/A	N/A			340	295	425	0	1 (PCI)	>100
Total (no DMA, with address/endian translations, with IPIF BarOffset reg, with DevNum reg.)	6	3	9	1	1	1	2480	1710	3230	2	2	>100
Total (with DMA, with address/endian translations with IPIF BarOffset reg, with DevNum reg.)	6	3	9	1	1	1	3025	2105	4155	2	2	>100
Total (with DMA, with address/endian translations without IPIF BarOffset reg, without DevNum reg.)	6	3	9	1	1	1	2815	1945	3890	2	2	>100
Total (with DMA, with address/endian translations)	6	3	9	1	1	0	2255	1625	3150	2	2	>100
Total (no DMA, with address/endian translations)	6	3	9	1	1	0	1565	1145	1995	2	2	>100
Total (no DMA, with address/endian translations, no configuration functionality)	6	3	9	0	1	0	1480	1055	1860	2	2	>100

Table 31: ***OPB PCI Bridge*** **FPGA Performance and Resource Utilization Benchmarks *(Contd)***

	Parameter Values						**Device Resources**					**f_{MAX}**
Total (no DMA, with address/endian translations)	1 req	3	9	N/A	0	1	1500	1055	1955	2	2	>100
Register bridge (no FIFOs) No interrupt or reset modules (no MIR/PENENCODE)	6	3	0	1	1	1	960	770	1160	0	2	>100
Register bridge (no FIFOs) No interrupt or reset modules (no MIR/PENENCODE)	6	31	0	0	1	1	820	695	1030	0	2	>100
Register bridge (no FIFOs) No interrupt or reset modules (no MIR/PENENCODE)	1	1	0	1	1	1	880	720	1115	0	2	>100
Register bridge (no FIFOs) No interrupt or reset modules (no MIR/PENENCODE)	1	1	0	0	1	1	742	640	930	0	2	>100
Register bridge (no FIFOs) No interrupt or reset modules (no MIR/PENENCODE)	1	1	0	0	0	1	600	530	745	0	2	>100
Register bridge (no FIFOs) No interrupt or reset modules (no MIR/PENENCODE)	1	1	0	N/A	1	0	445	415	480	0	2	>100

Notes:
1. These benchmark designs contain only the OPB PCI Bridge with registered inputs/outputs without any additional logic. Benchmark numbers approach the performance ceiling rather than representing performance under typical user conditions.
2. N/A - Not applicable

Reference Documents

The following documents contain reference information important to understanding the OPB PCI Bridge design:

- *Processor IP Reference Guide*
- *Xilinx LogiCORE PCI Interface v3.0 Product Specification*
- *Xilinx The Real-PCI Design Guide v3.0*
- *IBM 64-Bit On-Chip Peripheral Bus Architecture Specifications Version 2.0*

Revision History

Date	Version	Revision
6/22/05	1.0	Initial Xilinx release.
7/14/05	1.1	In Table 2, added PCI Bus as interface type for P28, P29, and P30; made minor typographical corrections.
12/12/05	1.2	Reorganized content, added dependencies tables, rewrote transaction descriptions, made extensive changes and rewrites in content.
1/25/06	1.3	Added Spartan-3 to supported devices and table footnote 1; added 3rd paragraph to Configuration Transactions section; in Table 1, changed Allowable Values for G63 to read 5 to 1.