

## Introduction

The On-Chip Peripheral Bus (OPB) to Processor Local Bus (PLB) Bridge module translates OPB transactions into PLB transactions. It functions as a slave on the OPB side and a master on the PLB side. Access to the control register and bus error status registers is user selectable from either the OPB or an optional DCR interface. The OPB to PLB Bridge is necessary in systems where an OPB master device, such as a DMA engine or an OPB based coprocessor, requires access to PLB devices (that is, high speed memory devices, etc.).

The Xilinx® OPB to PLB Bridge design allows customers to tailor the bridge to suit their application by setting certain parameters to enable / disable features. The parameterizable features of the design are discussed in **OPB to PLB Bridge Parameters**. Differences between the IBM OPB to PLB Bridge implementation and the Xilinx OPB to PLB Bridge implementation are also highlighted and explained.

In subsequent sections, the OPB to PLB Bridge, when convenient, is referred to as the Bridge In (BGI), and the PLB to OPB Bridge is referred to as the Bridge Out (BGO). This terminology reflects a PLB centric convention of data flowing “in from” and “out to” the peripheral bus, respectively. However, this is only a naming convention and in no way restricts the use of these bridges in alternative processor / bus configurations.

## Features

The Xilinx OPB to PLB Bridge is a soft IP core with the following features:

- 64-bit PLB Master interface
  - Communicates with 64-bit PLB slaves
  - 32-bit PLB devices are not supported
  - Non-burst transfers of 1 to 8 bytes
  - Burst transfers, including word and double-word bursts of fixed lengths, up to 16 words of data
  - Cacheline transactions of 4, 8, and 16 words

LogiCORE™ IP Facts		
Core Specifics		
See <a href="#">EDK Supported Device Families</a> .		
Version of core	opb2plb_bridge	v1.00c
Resources Used		
	Min	Max
I/O	390	390
LUTs	467	898
FFs	630	720
Block RAMs	0	0
Provided with Core		
Documentation	Product Specification	
Design File Formats	VHDL	
Constraints File	N/A	
Verification	N/A	
Instantiation Template	N/A	
Reference Designs	None	
Design Tool Requirements		
Xilinx Implementation Tools	See <a href="#">Tools</a> for requirements.	
Verification		
Simulation		
Synthesis		
Support		
Provided by Xilinx, Inc.		

- Translates OPB sequential accesses (bursts) to either cacheline or fixed length PLB burst transfers
  - Target word first order supported when cacheline transactions selected
  - Using PLB burst transfers yields better bus cycle efficiency but may increase logic utilization and degrade timing in the system
  - Performing only single beat and cacheline transactions reduces system logic utilization and improves timing through PLB slave IP simplification
- OPB Slave interface
  - 32-bit OPB Slave interface with byte enable transfers

**Note:** Does not support dynamic bus sizing or non-byte enable transactions

  - Decodes up to 4 separate address ranges
  - PLB and OPB clocks can have a 1:1, 2:1, 3:1, or 4:1 synchronous relationship (OPB clock frequency must be less than or equal to the PLB clock frequency)
  - Asserts BGI\_retry if the bridge is busy with a PLB transaction and a new OPB request is received.
- Bus Error Status Register (BESR) and Bus Error Address Register (BEAR) provide bus error status, and Bridge Control Register (BCR) provides bridge control functions
  - Parameterizable selection between DCR or OPB Slave interface, which provides access to BESR, BEAR, and BCR
- Posted write buffer 16 words deep
- Read prefetch buffer 16 words deep
- Edge-type interrupt generated when a bus error is detected

## High Level Description

**Figure 1** provides a high-level overview of the design of the OPB to PLB Bridge. The bridge reset signals (OPB\_Rst and PLB\_Rst) should be driven by a system level reset signal and should be active for a minimum of two OPB clocks to avoid any metastability problems between the OPB and PLB sides of the bridge.

OPB transactions are received and decoded in the OPB interface logic. On write transactions, write data is first queued up into a write buffer. The OPB interface logic then signals the PLB interface logic with the necessary information to begin the PLB transaction at the address specified.

On read transactions, read data from the PLB is streamed into a read buffer which the OPB interface logic can read and send to the OPB. When the PLB transaction is complete, the PLB interface logic signals this to the OPB interface so that another transaction can be performed. FIFOs and synchronizer logic are used to decouple the PLB and OPB logic to support asynchronous clock relationships between them. The interface to the error status registers is user selectable between the OPB slave interface or an optional DCR interface using a design parameter.

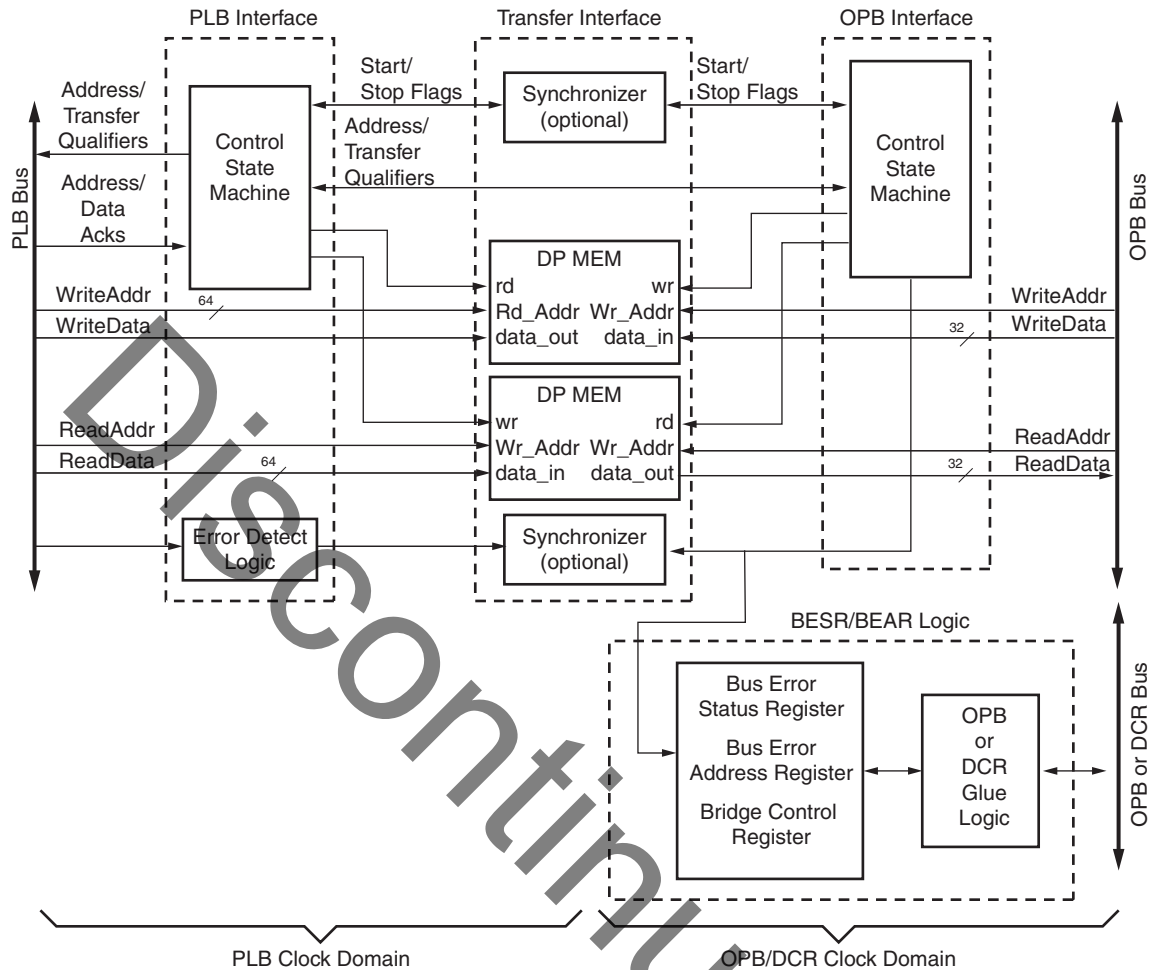


Figure 1: High Level Overview Of OPB To PLB Bridge

### OPB Interface

The OPB interface is designed with a pipelined architecture to improve timing and to support high clock frequencies. Input and Output signals to the OPB are designed to be driven through flip-flops for better timing. Pipelining introduces some additional latency in the design since some signals are delayed through registers. However, the use of pipelining balances transaction latency with higher clock frequencies.

The OPB slave interface is only designed to respond as a byte enable device. It does not support dynamic bus sizing transactions. For systems requiring a legacy OPB master to be able to talk to the bridge, an interface module called the OPB Byte Enable Interface (BEIF) is available to translate the transactions.

### Address Decode Cycle

OPB transactions begin with an address decode cycle. A design parameter allows the user to specify the number of address ranges the bridge will respond to and each range has parameters that specify the base address and the upper address for that range as well as a parameter that specifies whether cache line prefetching is allowed for reads from that range. There are also two parameters for each range that specify how writes can occur to that range. If bursts are enabled for a range then multidata writes will be handled using bursts on the PLB. If bursts are disabled for a range and line writes are enabled, then multidata writes will be handled using cache line transfers as long as enough write data exists in the write buffer to fill a cache line. The user can select the size of the read and write cache lines by setting two additional parameters.

An extra clock cycle is used to decode the address and transfer qualifiers of an OPB request before the address can be acknowledged.

The OPB Interface is designed to handle a single transaction at a time. It must complete a transaction before it will accept another transaction. While it is busy performing a transaction, it will assert its retry signal on any subsequent transactions.

A transaction consists of a single data transfer if `OPB_seqAddr` is not asserted. If `OPB_seqAddr` is asserted the bridge will attempt to minimize the number of PLB transactions necessary to complete the requested OPB transaction. Design parameters allow the user to specify the use of prefetching for read transactions, which may cause coherency problems or problems with destructive reads for some types of PLB slave devices (see Note 4 in Table 1). Additional design parameters allow the user to specify the use of bursts or cache line writes for write transactions.

Burst write data transfers will be terminated with `BGI_retry` when the internal write buffer is full (16 words of data).

## Deadlock Prevention

The OPB to PLB Bridge implements address decoders that map the complement of the OPB address range specified for the Bridge Out. The Bridge Out, when converting a PLB transaction to an OPB transaction, provides the mechanism for differentiating the OPB address space from the PLB address space (within the total address space) by using two generics (parameters) called `C_BASEADDR` and `C_HIGHADDR`. These parameters define that portion of the total address space that OPB slave peripherals will respond to. Therefore the complement of this address range defines that portion of the total address space that OPB slave peripherals will **not** respond to. The OPB to PLB Bridge accepts the same values for the `C_RNG0_BASEADDR` and `C_RNG0_HIGHADDR` parameters that the Bridge Out uses, but only converts transactions for addresses (specified by OPB masters) that **do not** correspond to the address range specified for OPB slaves, that is, the complement of the `C_BASEADDR` to `C_HIGHADDR` range. This allows OPB masters to perform transactions with peripherals or memory devices located in the PLB address space.<sup>1</sup>

To prevent deadlock<sup>2</sup> between the Bridge Out and the Bridge In, a `BGI_Trans_Abort` signal goes from the PLB to OPB Bridge (BGO) to the OPB to PLB Bridge (BGI). This signal causes the OPB to PLB Bridge to abort a PLB read transaction that is currently in the address phase and issue a retry on the OPB.

This forces the OPB to PLB Bridge to get off the OPB when the PLB to OPB Bridge is waiting for the bus, thus breaking the deadlock condition. If the current PLB transaction has completed the address phase but has not been promoted to the primary transaction, a deadlock condition can still occur. The OPB to PLB Bridge has a timeout counter that will start when `BGI_Trans_Abort` is detected and will timeout after `C_ABORT_DLY_CNT` OPB clocks. When this timeout occurs a retry will be issued on the OPB side to allow the deadlock to break, and when the PLB read transaction completes, the data is discarded.

`BGI_Trans_Abort` can only interrupt a read transaction. Since writes are fully buffered by the OPB to PLB Bridge, they do not cause deadlock. Also the PLB to OPB bridge buffers write data so it only needs to assert `BGI_Trans_Abort` when it has an OPB read pending.

1. In systems with multiple hierarchical bus structures, the sharing of `C_BASEADDR` and `C_HIGHADDR` by each pair of bridges simplifies memory space allocation, helping to avoid overlaps. Each pair of bridges is assigned unique values for their `C_BASEADDR` and `C_HIGHADDR` parameters that encompass the address range of the peripherals associated with that peripheral bus. It also ensures that any bus master can communicate with any peripheral slave anywhere in the system.
2. Deadlock can only occur when both bridges are attempting read transactions at the same time. This occurs because read transactions on the PLB may take longer than the length of an OPB timeout period, so the Bridge In must assert its timeout suppress signal whenever it is requested to do a read transaction. In addition there is no timeout on the PLB once the address phase of a transaction has completed. Therefore, if the Bridge Out is attempting a read transaction from the PLB to the OPB at the same time that the Bridge In is attempting a read transaction from the OPB to the PLB, deadlock can occur since the Bridge Out will be waiting for the OPB and the Bridge In will be waiting for the PLB. Since write transactions are "fire and forget", the Bridge In does not need to assert its timeout suppress signal, so deadlock cannot occur if the Bridge In is performing a write transaction.

## Write Transactions

On a write transaction from the OPB, the write data is directly written into a buffer. After the OPB write transaction is completed (or was terminated after 16 words because the buffer is full), the OPB interface logic signals to the PLB interface logic to carry out the transaction over the PLB.

It then provides the relevant information such as byte enables, and destination address. The OPB interface logic then waits for confirmation that the data transfer is complete before it can accept another OPB transaction. The OPB interface will issue a retry for any request it receives while it is waiting for the completion confirmation.

## Read Transactions

On a read transaction, data must be requested from the PLB. If the read is not a burst read (`OPB_seqAddr = 0`) then a single word of data is requested from the PLB to complete the transaction.

If the read is a burst read (`OPB_seqAddr = 1`) and prefetching is allowed (`C_RNGn_PREFETCH = 1`, `n = 0` through 3), then the PLB prefetches one cacheline of data. Since the OPB burst protocol does not provide burst length information, data must be prefetched in blocks to improve data transfer efficiency or else a series of single word reads is required.

Multiple read prefetches may be required for long read bursts. OPB masters must be careful when performing read bursts at address locations which have read side effects because of the prefetching feature (coherency or destructive read problems). Single beat reads should be used when accessing any special memory locations, such as peripherals that destroy the contents of a register when it is read.

## Burst Write Transactions

On a write burst from the OPB, the write data is directly written into a buffer. If the address is double word aligned, address bits 29 to 31 = '000' then after the OPB write transaction is completed (or was terminated after 16 words because the buffer is full), the OPB interface logic signals to the PLB interface logic to carry out the transaction as a burst over the PLB.

If the starting address is not double word aligned but is word aligned address bits 29 to 31 = '100', the bridge will convert the first word into a single transaction and force retries on the OPB until the first word has completed on the PLB. The remaining data can then be burst into the buffer until the OPB write transaction is completed (or was terminated after 16 words because the buffer is full), the OPB interface logic signals to the PLB interface logic to carry out the transaction as a burst over the PLB.

Bursts must start on a word aligned address, address bits 30 to 31 = '00', end on a word aligned address, address bits 30 to 31 = '00', and contain only full word data.

Other types of OPB bursts, such as bytes and half words are not supported by the OPB to PLB Bridge.

## Transfer Interface

The transfer interface facilitates the movement of data between the OPB and PLB interface logic which may be operating in different clock domains. Dual-port memory buffers are used as the mechanism to move write data and read data between OPB and PLB logic. Control signals are passed through synchronizing logic to handle the transition between OPB and PLB clock domains. The user has the option to specify either a synchronous or asynchronous timing relationship between the PLB and OPB clocks. If the clocks are known to be synchronous, then the user should set `C_OPB_PLB_SYNCH_CLKS` to 1 and much of the transfer interface logic will be simplified to reduce both logic utilization and latency.

## PLB Interface

The PLB interface is designed with a pipelined architecture to improve timing and to support high clock frequencies. Input and Output signals to the PLB are designed to be driven through flip-flops for better timing. Pipelining introduces some additional latency in the design since some signals are delayed through registers. However, the use of pipelining balances transaction latency with higher clock frequencies.

The PLB Interface logic initiates PLB read/write transactions as a PLB master to transfer data as requested by the OPB interface logic. The PLB interface logic for each address range can be instantiated in two types:

## Single Word and Line Transactions Only

In this mode, the PLB interface will perform only single word or line transactions as a 64 bit master. This allows the OPB to PLB Bridge to emit transactions similar to the type that a CPU emits (cacheline transactions). In this mode the C\_RNGn\_BURST (n = 0 to 3) parameters would all be set to zero and the C\_RNGn\_PREFETCH and C\_RNGn\_LINE parameters would be set to one.

Though this will reduce the efficiency of data transfers over the PLB, it may improve overall system performance and reduce logic utilization through the simplification of PLB slaves<sup>1</sup>. In this mode, cacheline data will be requested target word first to help reduce latency.

## Full Transactions

In this mode, the PLB interface will utilize single word transfers, line read transfers, or fixed length burst transfers. It only supports 64 bit PLB slaves. In this mode the C\_RNGn\_BURST and C\_RNGn\_PREFETCH parameters would be set to one and the C\_RNGn\_LINE parameters have no effect.

This mode offers more efficient use of the PLB. However, requiring PLB slaves to support burst protocols where they otherwise would not, may result in added complexity in the PLB slaves and a decrease in overall system performance (see the section [Single Word and Line Transactions Only](#)).

If an OPB burst is broken up into more than one transaction on the PLB, the PLB transactions are generated with the bus locked to maintain atomicity. If a burst transaction is terminated prematurely by the slave, then the master will try additional transactions (single beat) to transfer the remaining data.

The PLB interface will translate the state of the OPB busLock signal to the PLB. This allows a series of locked OPB transactions to generate a series of locked PLB transactions.

## Error Status Register with OPB or DCR Interface

PLB Read errors are passed back to the OPB along with the read data. The OPB reports this with the errAck flag. Write errors cannot be passed back to the OPB because the writes are posted and performed on the PLB after the OPB transaction has completed.

However, whenever a read or write error is detected, a bus error status register (BESR) is updated along with a bus error address register (BEAR) to record information about the error. The error status register can generate an interrupt request informing the OPB master and/or the CPU that an error has occurred.

The error status register can be configured to either lock or not lock the error flag through the Bridge Control Register (BCR). Locking errors means that the BESR and BEAR will not be updated on subsequent errors until the error bit has been cleared.

Unlocking the error registers allows future error conditions to overwrite the BESR and BEAR. If C\_LCK\_LOCKS\_INT is set to a 1 then locking the BESR and BEAR will also lock the interrupt request so that no additional error interrupts are generated until the error has been cleared. Setting C\_LCK\_LOCKS\_INT to a 0 will allow additional error interrupts to be generated even though the BESR and BEAR are locked. However Interrupt requests are only generated if the IRE bit in the BCR is set to a '1'.

A parameter selects whether the OPB slave interface is used, or an optional DCR interface is provided, to access the error status registers. Selecting an OPB interface allows the OPB master that initiated the transaction to read the error status and address directly, using an OPB transaction.

Selecting a DCR interface requires a DCR master to control the access of error information. In the context of this document a DCR master can be either a DCR equipped CPU, or a processor utilizing an OPB to DCR bridge. If the DCR master did not initiate the transaction that caused the error, it may need additional information from the OPB master that did, before corrective action can be taken. The DCR interface, if selected, operates in the same clock domain as the OPB interface. If the DCR interface is present then the registers are not available from the OPB interface.

Additional information about the error can be obtained from the error status registers in the PLB slaves. PLB slaves that are capable of generating an error typically also implement a set of status registers.

1. In an FPGA implementation of a PLB equipped system, the overall system performance can be increased and the logic resources reduced if PLB slaves do not include support for burst modes. Some processors do not utilize burst transactions, so in those cases the resource savings and performance improvements gained by simplifying the slaves can significantly improve a system design.

## OPB to PLB Bridge Parameters

To allow the user to obtain an OPB to PLB Bridge that is uniquely tailored for their system, certain features are parameterizable in the Xilinx OPB to PLB Bridge design. This allows the user to have a design that only utilizes the resources required by their system and runs at the best possible performance. The features that are parameterizable in the Xilinx OPB to PLB Bridge design are shown in [Table 1](#).

**Table 1: OPB to PLB Bridge Design Parameters**

Grouping/Number	Feature / Description	Parameter Name	Allowable Values	Default Value	VHDL Type	
Bridge Features	G1	PLB and OPB clocks are the same frequency <sup>(1)</sup>	C_SAME_CLKS	0 (false) or non-zero (true)	1	integer
	G2	Size of read prefetch cache lines	C_RD_PREFETCH_CNT	4, 8, or 16	8	integer
	G3	Size of write cache lines	C_WR_CACHELINE_SIZE	4, 8, or 16	8	integer
	G4	Error register access	C_OPB_REG_INTFC	1 = OPB interface access 0 = DCR interface access	1	integer
	G5	Number of OPB clocks before abort	C_ABORT_DLY_CNT	10	10	integer
	G6	OPB address bus width	C_OPB_AWIDTH	32	32	integer
	G7	PLB address bus width	C_PLB_AWIDTH	32	32	integer
	G8	Target device family (not utilized in this version of the bridge)	C_FAMILY	"virtex2p"	"virtex2p"	string
	G9	Number of address ranges present	C_NUM_ADDR_RNG	1 to 4	1	integer
OPB Slave Interface	G10	Range 0 Base Address <sup>(2)</sup>	C_RNG0_BASEADDR	Valid Address Range	none <sup>(3)</sup>	std_logic_vector
	G11	Range 0 High Address <sup>(2)</sup>	C_RNG0_HIGHADDR	Valid Address Range	none <sup>(3)</sup>	std_logic_vector
	G12	Range 1 Base Address <sup>(2)</sup>	C_RNG1_BASEADDR	Valid Address Range	none <sup>(3)</sup>	std_logic_vector
	G13	Range 1 High Address <sup>(2)</sup>	C_RNG1_HIGHADDR	Valid Address Range	none <sup>(3)</sup>	std_logic_vector
	G14	Range 2 Base Address <sup>(2)</sup>	C_RNG2_BASEADDR	Valid Address Range	none <sup>(3)</sup>	std_logic_vector
	G15	Range 2 High Address <sup>(2)</sup>	C_RNG2_HIGHADDR	Valid Address Range	none <sup>(3)</sup>	std_logic_vector
	G16	Range 3 Base Address <sup>(2)</sup>	C_RNG3_BASEADDR	Valid Address Range	none <sup>(3)</sup>	std_logic_vector
	G17	Range 3 High Address <sup>(2)</sup>	C_RNG3_HIGHADDR	Valid Address Range	none <sup>(3)</sup>	std_logic_vector

Table 1: OPB to PLB Bridge Design Parameters (Continued)

Grouping/Number	Feature / Description	Parameter Name	Allowable Values	Default Value	VHDL Type	
OPB Slave Interface	G18	Range 0 Prefetchable <sup>(4)</sup>	C_RNG0_PREFETCH	1 = allow prefetching 0 = no prefetching	0	integer
	G19	Range 1 Prefetchable <sup>(4)</sup>	C_RNG1_PREFETCH	1 = allow prefetching 0 = no prefetching	0	integer
	G20	Range 2 Prefetchable <sup>(4)</sup>	C_RNG2_PREFETCH	1 = allow prefetching 0 = no prefetching	0	integer
	G21	Range 3 Prefetchable <sup>(4)</sup>	C_RNG3_PREFETCH	1 = allow prefetching 0 = no prefetching	0	integer
	G22	Range 0 supports Bursts	C_RNG0_BURST	1 = allow bursts 0 = no bursts	0	integer
	G23	Range 1 supports Bursts	C_RNG1_BURST	1 = allow bursts 0 = no bursts	0	integer
	G24	Range 2 supports Bursts	C_RNG2_BURST	1 = allow bursts 0 = no bursts	0	integer
	G25	Range 3 supports Bursts	C_RNG3_BURST	1 = allow bursts 0 = no bursts	0	integer
	G26	Range 0 supports cache line writes	C_RNG0_LINE	1 = allow cache line writes 0 = no cache line writes	0	integer
	G27	Range 1 supports cache line writes	C_RNG1_LINE	1 = allow cache line writes 0 = no cache line writes	0	integer
	G28	Range 2 supports cache line writes	C_RNG2_LINE	1 = allow cache line writes 0 = no cache line writes	0	integer
	G29	Range 3 supports cache line writes	C_RNG3_LINE	1 = allow cache line writes	0	integer
	G30	OPB Register Base Address <sup>(5)</sup>	C_OPB_BASEADDR	Valid Address Range	none <sup>(6)</sup>	std_logic_vector
	G31	OPB Register High Address <sup>(5)</sup>	C_OPB_HIGHADDR	Valid Address Range	none <sup>(6)</sup>	std_logic_vector
G32	OPB Data Bus Width	C_OPB_DWIDTH	32	32	integer	
PLB Master Interface	G33	PLB Data Bus Width	C_PLB_DWIDTH	64	64	integer
	G34	PLB Master Priority	C_PLB_PRIORITY	0, 1, 2, or 3	0	integer

**Table 1: OPB to PLB Bridge Design Parameters (Continued)**

Grouping/Number		Feature / Description	Parameter Name	Allowable Values	Default Value	VHDL Type
DCR Interface	G35	DCR Base Address <sup>(7)</sup>	C_DCR_BASEADDR	Valid DCR address	none <sup>(8)</sup>	std_logic_vector
	G36	DCR High Address <sup>(7)</sup>	C_DCR_HIGHADDR	Valid DCR address	none <sup>(8)</sup>	std_logic_vector
	G37	DCR Address Bus Width	C_DCR_AWIDTH	10	10	integer
	G38	DCR Data Bus Width	C_DCR_DWIDTH	32	32	integer

Discontinued IP

Table 1: OPB to PLB Bridge Design Parameters (Continued)

Grouping/Number	Feature / Description	Parameter Name	Allowable Values	Default Value	VHDL Type
Miscellaneous	G39	Active interrupt state <sup>(9)</sup>	'1' = rising edge '0' = falling edge	'1'	std_logic
	G40	Software reset count	2	2	integer
	G25	LCK bit locks interrupt	1 = LCK bit in BESR locks interrupt request also  0 = LCK bit in BESR has no effect on interrupt request	1	integer

**Notes:**

- Setting this parameter to 1 causes the bridge to be built without synchronization registers between the OPB side and the PLB side. If the PLB is run at a different clock speed than the OPB, this parameter should be set to 0 so that these synchronization registers are included in the bridge. Without them there is a good chance that the internal control signals will be misinterpreted and the bridge will fail to complete its transactions, causing the system to hang.
- C\_RNG0\_BASEADDR and C\_RNG0\_HIGHADDR should be set to the same values used for the PLB to OPB Bridge (Bridge Out). The OPB to PLB Bridge (Bridge In) will decode addresses based on the inverse of these parameters such that the Bridge In will respond to all addresses that the Bridge Out does not. Each of the ranges specified by C\_RNG0\_BASEADDR to C\_RNG0\_HIGHADDR, C\_RNG1\_BASEADDR to C\_RNG1\_HIGHADDR, C\_RNG2\_BASEADDR to C\_RNG2\_HIGHADDR, and C\_RNG3\_BASEADDR to C\_RNG3\_HIGHADDR should each comprise a complete, contiguous power of two range such that  $\text{range}_0 = 2^m$ , with the m least significant bits of C\_RNG0\_BASEADDR set to zero;  $\text{range}_1 = 2^n$ , with the n least significant bits of C\_RNG1\_BASEADDR set to zero;  $\text{range}_2 = 2^p$ , with the p least significant bits of C\_RNG2\_BASEADDR set to zero; and  $\text{range}_3 = 2^q$ , with the q least significant bits of C\_RNG3\_BASEADDR set to zero. These address ranges specify the valid addresses the OPB to PLB Bridge will respond to. They should map to valid PLB slave peripheral addresses. These ranges must be non-overlapping and should not coincide or overlap any address ranges specified for the associated PLB to OPB Bridge.
- There is no default value for C\_RNG0\_BASEADDR through C\_RNG3\_BASEADDR or C\_RNG0\_HIGHADDR through C\_RNG3\_HIGHADDR to ensure the user specifies these values. If the user fails to specify these values for any range that is utilized by the bridge, a compiler error will result.
- If C\_RNGn\_PREFETCH (n is 0 to 3) is set to a 1 the bridge will attempt to fill its read buffer using cache line reads whenever an OPB master is requesting sequential address read transactions (OPB read bursts) to this address range, in an attempt to minimize PLB transactions. However, prefetching can cause coherency problems and problems with destructive read devices. Coherency problems will occur if a different PLB master writes new data to a location residing within a cache line following a read of that cache line by the bridge. During the following sequential OPB reads the OPB master would receive stale data from the bridge. A different problem occurs when reads from the PLB slave being accessed are destructive. Prefetching may cause the bridge to consume more data than was intended for the OPB transaction, leaving the data unavailable for the transaction it was intended for. These problems can be alleviated if OPB masters do not perform sequential reads from PLB slaves that exhibit these conditions. Alternatively, this parameter can be set to 0 and the bridge will only perform single beat read transactions on the PLB in this address range, regardless of whether or not the OPB master is requesting sequential read transactions. This parameter defaults to 0.
- As a minimum, C\_OPB\_BASEADDR must have the least significant 4 bits set to zero. C\_OPB\_HIGHADDR must be at least C\_OPB\_BASEADDR + 31. These parameters are used to determine the number of most significant address bits the bridge uses to decode its OPB register address space. These parameters allow the user to trade-off address space resolution with size and speed of the bridge. If the user specifies a value for C\_OPB\_HIGHADDR that is greater than the minimum value, the range specified by C\_OPB\_BASEADDR and C\_OPB\_HIGHADDR should encompass a complete, contiguous power of two range (that is,  $\text{range} = 2^n$ , with the n least significant bits of C\_OPB\_BASEADDR set to zero). For example, if C\_OPB\_BASEADDR is set to 1000000h, then legal values for C\_OPB\_HIGHADDR are 100001Fh through 1FFFFFFh. The number of upper address bits decoded to select the bridge would be 27 for the first case or 4 for the latter. The first case results in the most efficient use of address space at the expense of additional FPGA resources, and reduces the maximum frequency at which the bridge can operate. The latter case uses fewer FPGA resources which may also result in a faster implementation, but the bridge registers are aliased within those 268,435,456 locations, preventing the use of those 268,435,444 redundant locations of address space for anything else. These parameters are ignored when C\_OPB\_REG\_INTFC is set to 0.
- There is no default value for C\_OPB\_BASEADDR or C\_OPB\_HIGHADDR to ensure the user specifies these values. If the user fails to specify these values a compiler error will result. If C\_OPB\_REG\_INTFC is set to 0 then these parameters should be set to "X'00000000".
- As a minimum, C\_DCR\_BASEADDR must have the two least significant bits set to a zero. C\_DCR\_HIGHADDR must be at least C\_DCR\_BASEADDR + 3 (since DCR addresses are always word addresses), and the two least significant bits must be set to a one. These parameters are used to determine the number of most significant address bits the bridge uses to decode its DCR register address space. These parameters allow the user to trade-off DCR address space resolution with size and speed of the bridge. If the user specifies a value for C\_DCR\_HIGHADDR that is greater than the minimum value, the range specified by C\_DCR\_BASEADDR and C\_DCR\_HIGHADDR should encompass a complete, contiguous power of two range (that is,  $\text{range} = 2^n$ , with the n least significant bits of C\_DCR\_BASEADDR set to zero). For example, if C\_DCR\_BASEADDR is set to 010h, then legal values for C\_DCR\_HIGHADDR are 013h, 017, or 01Fh. The number of upper address bits decoded to select the bridge would be 8, 7, or 6, respectively. The first case results in the most efficient use of DCR address space at the expense of additional FPGA resources and possibly reduces the maximum frequency at which the bridge can operate. The last case uses fewer FPGA resources which may also result in a faster implementation, but the bridge registers are aliased within those 16 locations, preventing the use of those redundant 13 locations of DCR address space for anything else. These parameters are ignored when C\_OPB\_REG\_INTFC is set to 1.
- There is no default value for C\_DCR\_BASEADDR or C\_DCR\_HIGHADDR to ensure the user specifies these values. If the user fails to specify these values a compiler error will result. If C\_OPB\_REG\_INTFC is set to 1 then these parameters should be set to "000000000".
- The interrupt request output (Bus\_Error\_Det, P6) is generated as an edge type interrupt request. C\_IRQ\_ACTIVE specifies the active state of the interrupt request given its type. If C\_IRQ\_ACTIVE = '1' the interrupt request will be generated as a rising edge and if C\_IRQ\_ACTIVE = '0' then a falling edge interrupt will be generated. The interrupt request output is a pulse, one OPB clock period long, with a polarity such that the leading edge of the pulse provides the specified type of edge interrupt output.

## Allowable Parameter Combinations

All combinations of OPB to PLB Bridge parameters are allowed; however, some parameters can cause other parameters to be irrelevant. See the Parameter - Port Dependencies section.

Also if a DCR bus is selected and if the width of the OPB address bus is greater than the width of the DCR data bus, reading the Bus Error Address Register (BEAR) will only return that portion of the most significant bits of the offending OPB address that will fit into a DCR data word.

## OPB to PLB Bridge I/O Signals

The I/O signals for the OPB to PLB Bridge are listed in [Table 2](#). The interfaces referenced in this table are shown in [Figure 1](#) in the OPB to PLB Bridge block diagram.

Table 2: OPB to PLB Bridge I/O Signals

Grouping		Signal Name	Interface	I/O	Description	Page
System	P1	OPB_Clk	System	I	OPB Clock	
	P2	PLB_Clk	System	I	PLB Clock	
	P3	OPB_Rst	System	I	OPB Reset <sup>(1)</sup>	
	P4	PLB_Rst	System	I	PLB Reset <sup>(1)</sup>	
	P5	BGI_Trans_Abort	System	I	BGI Transaction Abort	
Interrupt Request	P6	Bus_Error_Det	BEAR / BESR Logic	O	PLB Bus error detected	
OPB Slave Signals	P7	OPB_select	OPB	I	OPB Select	
	P8	OPB_ABus	OPB	I	OPB Address Bus	
	P9	OPB_RNW	OPB	I	OPB Read Not Write	
	P10	OPB_BE	OPB	I	OPB Byte Enables	
	P11	OPB_DBus	OPB	I	OPB Data Bus	
	P12	OPB_seqAddr	OPB	I	OPB Sequential Address	
	P13	BGI_DBus	OPB	O	Bridge Data Bus	
	P14	BGI_retry	OPB	O	Bridge Retry	
	P15	BGI_toutSup	OPB	O	Bridge Timeout Suppress	
	P16	BGI_xferAck	OPB	O	Bridge Transfer Acknowledge	
	P17	BGI_errAck	OPB	O	Bridge Error Acknowledge	

Table 2: OPB to PLB Bridge I/O Signals (Continued)

Grouping	Signal Name	Interface	I/O	Description	Page
PLB Master Signals	P18	BGI_request	PLB	O	Bridge In Bus Transaction Request
	P19	BGI_ABus	PLB	O	Bridge In Address Bus
	P20	BGI_RNW	PLB	O	Bridge In Read Not Write
	P21	BGI_BE	PLB	O	Bridge In Byte Enables
	P22	BGI_size	PLB	O	Bridge In Transfer Size
	P23	BGI_type	PLB	O	Bridge In Transfer Type
	P24	BGI_priority	PLB	O	Bridge In Transaction Priority
	P25	BGI_rdBurst	PLB	O	Bridge In Read Burst Request
	P26	BGI_wrBurst	PLB	O	Bridge In Write Burst Request
	P27	BGI_busLock	PLB	O	Bridge In Bus Lock Request
	P28	BGI_abort	PLB	O	Bridge In Abort Request
	P29	BGI_lockErr	PLB	O	Bridge In Lock Error Status
	P30	BGI_mSize	PLB	O	Bridge In Data Bus Size
	P31	BGI_ordered	PLB	O	Bridge In Ordered Transfer Indicator
	P32	BGI_compress	PLB	O	Bridge In Compressed Data Transfer Indicator
	P33	BGI_guarded	PLB	O	Bridge In Guarded Memory Access Indicator
	P34	BGI_wrDBus	PLB	O	Bridge In Write Data Bus
	P35	PLB_RdWdAddr	PLB	I	PLB Read Word Address Indicators
	P36	PLB_RdDBus	PLB	I	PLB Read Data Bus
	P37	PLB_AddrAck	PLB	I	PLB Address Acknowledge
	P38	PLB_RdDAck	PLB	I	PLB Read Data Acknowledge
	P39	PLB_WrDAck	PLB	I	PLB Write Data Acknowledge
	P40	PLB_Rearbitrate	PLB	I	PLB Rearbitrate Request
	P41	PLB_Busy	PLB	I	PLB Slave Busy Indicator
	P42	PLB_Err	PLB	I	PLB Slave Error Indicator
	P43	PLB_RdBTerm	PLB	I	PLB Read Burst Termination Request
	P44	PLB_WrBTerm	PLB	I	PLB Write Burst Termination Request
	P45	PLB_SSize	PLB	I	PLB Slave Size
	P46	PLB_pendReq	PLB	I	PLB Pending Request Indicator
	P47	PLB_pendPri	PLB	I	PLB Pending Request Priority Indicator
P48	PLB_reqPri	PLB	I	PLB Current Request Priority	

**Table 2: OPB to PLB Bridge I/O Signals (Continued)**

Grouping	Signal Name	Interface	I/O	Description	Page
DCR Signals	P49	DCR_Read	DCR	I	DCR Read Request
	P50	DCR_Write	DCR	I	DCR Write Request
	P51	DCR_ABus	DCR	I	DCR Address Bus
	P52	DCR_DBus	DCR	I	DCR Data Bus (out of the processor)
	P53	BGI_dcr_DBus	DCR	O	Bridge In Data Bus (into the processor)
	P54	BGI_Ack	DCR	O	Bridge In Transfer Acknowledge

**Notes:**

- The bridge reset inputs must be active for a minimum of two OPB clock periods. PLB\_Rst and OPB\_Rst are or'd together, along with the software reset, to produce the internal bridge reset signal.

## Parameter-Port Dependencies

The width of many of the OPB to PLB Bridge signals depends on the width of the various address and data buses. In addition, when certain features are parameterized away, the related input signals are unused and the related output signals are set to constant values. The dependencies between the OPB to PLB Bridge design parameters and I/O signals are shown in [Table 3](#).

**Table 3: Parameter - Port Dependencies**

Grouping	Name	Affects	Depends	Relationship Description	
Design Parameters	G1	C_SAME_CLKS			
	G2	C_RD_PREFETCH_CNT			
	G3	C_WR_CACHELINE_SIZE			
	G4	C_OPB_REG_INTFC	P49 through P54		If C_OPB_REG_INTFC = 0 then the DCR interface input signals are not connected to any logic and the DCR interface output signals are tied to their inactive state.
	G5	C_ABORT_DLY_CNT			
	G6	C_OPB_AWIDTH	G10 through G17, G30, G31, P8	G7	Width of RNGn and OPB BASEADDR and HIGHADDR parameter pairs. OPB and PLB address buses must be same width.
	G7	C_PLB_AWIDTH	P19	G6	OPB and PLB address buses must be same width.
	G8	C_FAMILY			
	G9	C_NUM_ADDR_RNG	G10 through G29		This parameter determines the number of RNGn BASEADDR and HIGHADDR parameter pairs used for OPB address decoding, starting from RNG0, and the number of corresponding RNGn PREFETCH, BURST, and LINE parameters that are used.

Table 3: Parameter - Port Dependencies (Continued)

Grouping	Name	Affects	Depends	Relationship Description
G10	C_RNG0_BASEADDR	G11	G6	The range specified by C_RNG0_BASEADDR and C_RNG0_HIGHADDR should span a complete, contiguous power of two address range. The width of C_RNG0_BASEADDR must be the same as C_OPB_AWIDTH.
G11	C_RNG0_HIGHADDR		G6, G10	The range specified by C_RNG0_BASEADDR and C_RNG0_HIGHADDR should span a complete, contiguous power of two address range. The width of C_RNG0_HIGHADDR must be the same as C_OPB_AWIDTH.
G12	C_RNG1_BASEADDR	G13	G6	The range specified by C_RNG1_BASEADDR and C_RNG1_HIGHADDR should span a complete, contiguous power of two address range. The width of C_RNG1_BASEADDR must be the same as C_OPB_AWIDTH.
G13	C_RNG1_HIGHADDR		G6, G12	The range specified by C_RNG1_BASEADDR and C_RNG1_HIGHADDR should span a complete, contiguous power of two address range. The width of C_RNG1_HIGHADDR must be the same as C_OPB_AWIDTH.
G14	C_RNG2_BASEADDR	G15	G6	The range specified by C_RNG2_BASEADDR and C_RNG2_HIGHADDR should span a complete, contiguous power of two address range. The width of C_RNG2_BASEADDR must be the same as C_OPB_AWIDTH.
G15	C_RNG2_HIGHADDR		G6, G14	The range specified by C_RNG2_BASEADDR and C_RNG2_HIGHADDR should span a complete, contiguous power of two address range. The width of C_RNG2_HIGHADDR must be the same as C_OPB_AWIDTH.
G16	C_RNG3_BASEADDR	G17	G6	The range specified by C_RNG3_BASEADDR and C_RNG3_HIGHADDR should span a complete, contiguous power of two address range. The width of C_RNG3_BASEADDR must be the same as C_OPB_AWIDTH.
G17	C_RNG3_HIGHADDR		G6, G16	The range specified by C_RNG3_BASEADDR and C_RNG3_HIGHADDR should span a complete power of two address range. The width of C_RNG3_HIGHADDR must be the same as C_OPB_AWIDTH.
G18	C_RNG0_PREFETCH			
G19	C_RNG1_PREFETCH			
G20	C_RNG2_PREFETCH			
G21	C_RNG3_PREFETCH			
G22	C_RNG0_BURST		G26	If both bursts and line writes are enabled for this range, bursts take precedence

**Table 3: Parameter - Port Dependencies (Continued)**

Grouping	Name	Affects	Depends	Relationship Description
G23	C_RNG1_BURST		G27	If both bursts and line writes are enabled for this range, bursts take precedence.
G24	C_RNG2_BURST		G28	If both bursts and line writes are enabled for this range, bursts take precedence.
G25	C_RNG3_BURST		G29	If both bursts and line writes are enabled for this range, bursts take precedence.
G26	C_RNG0_LINE	G22		If bursts are enabled for this range this parameter is ignored.
G27	C_RNG1_LINE	G23		If bursts are enabled for this range this parameter is ignored.
G28	C_RNG2_LINE	G24		If bursts are enabled for this range this parameter is ignored.
G29	C_RNG3_LINE	G25		If bursts are enabled for this range this parameter is ignored.
G30	C_OPB_BASEADDR	G31	G4, G6	The range specified by C_OPB_BASEADDR and C_OPB_HIGHADDR should span a complete power of two address range. The width of C_OPB_BASEADDR must be the same as C_OPB_AWIDTH. Not used if C_OPB_REG_INTFC = 0.
G31	C_OPB_HIGHADDR		G4, G6, G30	The range specified by C_OPB_BASEADDR and C_OPB_HIGHADDR should span a complete power of two address range. The width of C_OPB_HIGHADDR must be the same as C_OPB_AWIDTH. Not used if C_OPB_REG_INTFC = 0.
G32	C_OPB_DWIDTH	P10, P11, P13		Width of OPB data buses. Eight (8) times the width of OPB byte enable bus (BE).
G33	C_PLB_DWIDTH	P21, P34, P36		Width of PLB data buses. Eight (8) times the width of PLB byte enable bus (BE).
G34	C_PLB_PRIORITY			
G35	C_DCR_BASEADDR	G36	G4, G37	The range specified by C_DCR_BASEADDR and C_DCR_HIGHADDR should span a complete power of two address range. The width of C_DCR_BASEADDR must be the same as C_DCR_AWIDTH. Not used if C_OPB_REG_INTFC = 1.
G36	C_DCR_HIGHADDR		G4, G35, G37	The range specified by C_DCR_BASEADDR and C_DCR_HIGHADDR should span a complete power of two address range. The width of C_DCR_HIGHADDR must be the same as C_DCR_AWIDTH. Not used if C_OPB_REG_INTFC = 1.

Table 3: Parameter - Port Dependencies (Continued)

Grouping	Name	Affects	Depends	Relationship Description	
	G37 C_DCR_AWIDTH	G35, G36, P51	G4	Width of DCR BASEADDR and HIGHADDR parameter pair. Width of DCR address bus. Not used if C_OPB_REG_INTFC = 1.	
	G38 C_DCR_DWIDTH	P52, P53	G4	Width of DCR data buses. Not used if C_OPB_REG_INTFC = 1.	
	G39 C_IRQ_ACTIVE	P6			
	G40 C_SWR_COUNT				
	G25 C_LCK_LOCKS_INT				
I/O Signals	P1 OPB_Clk				
	P2 PLB_Clk				
	P3 OPB_Rst				
	P4 PLB_Rst				
	P5 BGI_Trans_Abort				
	P6 Bus_Error_Det		G25	LCK bit in BESR affects when new interrupt request is generated.	
	P7 OPB_select				
	P8 OPB_ABus(0:C_OPB_AWIDTH - 1)		G6	Width set by C_OPB_AWIDTH.	
	P9 OPB_RNW				
	P10 OPB_BE(0:C_OPB_DWIDTH / 8 - 1)		G32	Width is 1/8 of C_OPB_DWIDTH.	
	P11 OPB_DBus(0:C_OPB_DWIDTH - 1)		G32	Width set by C_OPB_DWIDTH.	
	P12 OPB_seqAddr				
	P13 BGI_DBus(0:C_OPB_DWIDTH - 1)		G32	Width set by C_OPB_DWIDTH.	
	P14 BGI_retry				
	P15 BGI_toutSup				
	P16 BGI_xferAck				
	P17 BGI_errAck				
	P18 BGI_request				
	P19 BGI_ABus(0:C_PLB_AWIDTH - 1)			G7	Width set by C_PLB_AWIDTH.
	P20 BGI_RNW				
	P21 BGI_BE(0:C_PLB_DWIDTH / 8 - 1)			G33	Width set by C_PLB_DWIDTH.
	P22 BGI_size(0:3)				
	P23 BGI_type(0:2)				
	P24 BGI_priority(0:1)				

**Table 3: Parameter - Port Dependencies (Continued)**

Grouping	Name	Affects	Depends	Relationship Description
P25	BGI_rdBurst			
P26	BGI_wrBurst			
P27	BGI_busLock			
P28	BGI_abort			
P29	BGI_lockErr			
P30	BGI_mSize(0:1)			
P31	BGI_ordered			
P32	BGI_compress			
P33	BGI_guarded			
P34	BGI_wrDBus(0:C_PLB_DWIDTH - 1)		G33	Width set by C_PLB_DWIDTH.
P35	PLB_RdWdAddr(0:3)			
P36	PLB_RdDBus(0:C_PLB_DWIDTH - 1)		G33	Width set by C_PLB_DWIDTH.
P37	PLB_AddrAck			
P38	PLB_RdDAck			
P39	PLB_WrDAck			
P40	PLB_Rearbitrate			
P41	PLB_Busy			
P42	PLB_Err			
P43	PLB_RdBTerm			
P44	PLB_WrBTerm			
P45	PLB_SSize(0:1)			
P46	PLB_pendReq			
P47	PLB_pendPri(0:1)			
P48	PLB_reqPri(0:1)			
P49	DCR_Read		G4	This signal is unconnected if C_OPB_REG_INTFC = 1.
P50	DCR_Write		G4	This signal is unconnected if C_OPB_REG_INTFC = 1.
P51	DCR_ABus(0:C_DCR_AWIDTH - 1)		G4, G37	This signal is unconnected if C_OPB_REG_INTFC = 1. Width set by C_DCR_AWIDTH.



Table 5: Bus Error Status Register Bit Definitions

Bit(s)	Name	Access	Reset Value	Description
0	Error Flag	Read Only	'0'	<b>ERR</b> : Indicates a bus error has occurred. '0' - No errors '1' - Error detected
1	Read / Write Status Flag	Read Only	'0'	<b>R / W</b> : Indicates a read or write transaction error. '0' - Transaction causing error was a write '1' - Transaction causing error was a read
2 to 15	Unused	Read Only	'0'	Unused
16	Byte Enable Bit 0	Read Only	'0'	<b>BE<sub>0</sub></b> : Indicates the status of PLB Byte Enable Bit 0 (the most significant bit) when <b>ERR = '1'</b>
17	Byte Enable Bit 1	Read Only	'0'	<b>BE<sub>1</sub></b> : Indicates the status of PLB Byte Enable Bit 1 when <b>ERR = '1'</b>
18	Byte Enable Bit 2	Read Only	'0'	<b>BE<sub>2</sub></b> : Indicates the status of PLB Byte Enable Bit 2 when <b>ERR = '1'</b>
19	Byte Enable Bit 3	Read Only	'0'	<b>BE<sub>3</sub></b> : Indicates the status of PLB Byte Enable Bit 3 when <b>ERR = '1'</b>
20	Byte Enable Bit 4	Read Only	'0'	<b>BE<sub>4</sub></b> : Indicates the status of PLB Byte Enable Bit 4 when <b>ERR = '1'</b>
21	Byte Enable Bit 5	Read Only	'0'	<b>BE<sub>5</sub></b> : Indicates the status of PLB Byte Enable Bit 5 when <b>ERR = '1'</b>
22	Byte Enable Bit 6	Read Only	'0'	<b>BE<sub>6</sub></b> : Indicates the status of PLB Byte Enable Bit 6 when <b>ERR = '1'</b>
23	Byte Enable Bit 7	Read Only	'0'	<b>BE<sub>7</sub></b> : Indicates the status of PLB Byte Enable Bit 7 (the least significant bit) when <b>ERR = '1'</b>
24 to 31	Reserved	Read Only	'0'	These bits are reserved for future use

### Bus Error Address Register (BEAR)

The following Bus Error Address Register contains the address of the data transfer that caused the error.

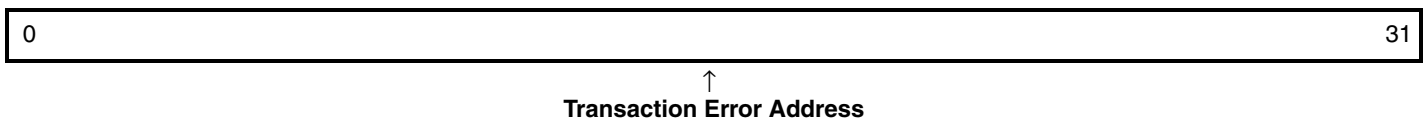


Figure 3: BEAR — Bus Error Address Register

## Bridge Control Register (BCR)

The following Bridge Control Register contains control bits which are defined in [Figure 4](#).

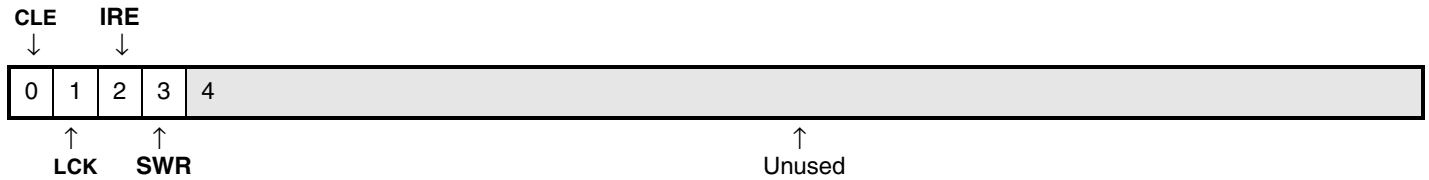


Figure 4: BCR — Bridge Control Register

Table 6: Bridge Control Register Bit Definitions

Bit(s)	Name	Access	Reset Value	Description
0	Clear Error	Write Only	n/a	<b>CLE:</b> Used to clear the bus error indicator flag in the BESR (note: when the BCR is read, this bit will always be zero) '0' no effect '1' reset error flag
1	Lock Control	Read/Write	'0'	<b>LCK:</b> Used to prevent BESR / BEAR updates after error detected. '0' - Registers overwritten for each new error '1' - Lock registers on first error until error flag is cleared.
2	Interrupt Request Enable	Read/Write	'1'	<b>IRE:</b> Enable or disable interrupt request output '0' - Disable interrupt request output '1' - Enable interrupt request output
3	Software Reset	Write Only	n/a	<b>SWR:</b> Allows processor to reset bridge (note: when the BCR is read, this bit will always be zero) '0' - no effect '1' - reset bridge
4 to 31	Unused	Read	'0'	Unused

## OPB to PLB Bridge Interrupt Descriptions

### Interrupts

The OPB to PLB Bridge has one edge (pulse) type interrupt request output called `Bus_Error_Det`. The active state (rising/falling) of the interrupt request is set by the user through the `C_IRQ_ACTIVE` generic (parameter).

Edge type interrupt requests are automatically reset to the inactive state on the next clock cycle and so do not require an explicit interrupt acknowledge.

By default, the interrupt output is enabled and each new error will generate a new interrupt request. If `C_LCK_LOCKS_INT` is set to a 1, the interrupt request output is also controlled by the LCK bit in the BCR. If the LCK bit is set to a '1', any additional error conditions will not generate a new interrupt request until the error flag has been cleared by writing a '1' to the CLE bit in the BCR. If `C_LCK_LOCKS_INT` is set to a 0 or the LCK bit is set to a '0', then each new error condition will generate a new interrupt request.

In all cases the Interrupt Request Enable (IRE) bit in the BCR can be used to enable / disable the interrupt request output.

The OPB to PLB Bridge also has one interrupt input called `BGI_Trans_Abort`. The intent of this input is to prevent a deadlock condition between the OPB to PLB Bridge and its associated PLB to OPB Bridge.

Whenever this signal is active the Bridge In will issue a retry to any OPB master requesting read access to the PLB and, if a PLB read transaction is in progress, it will abort that transaction (see ["Deadlock Prevention" on page 4](#)).

## Design Implementation

### Device Utilization and Performance Benchmarks

Since the OPB to PLB Bridge is a module that will be used with other design pieces in the FPGA, the utilization and timing numbers reported in this section are just estimates. As the OPB to PLB Bridge is combined with other pieces of the FPGA design, the utilization of FPGA resources and timing of the OPB to PLB Bridge design will vary from the results reported here.

To analyze the OPB to PLB Bridge timing within the FPGA, a design was created that instantiated the OPB to PLB Bridge with registers on all of the inputs and outputs. This allowed a constraint to be placed on the clock net to yield more realistic timing results. The  $f_{MAX}$  parameter specified in Table 7 reflects the use of this design for timing measurements. Note, however that the resource utilization values reported in Table 7 do not include these registers.

The OPB to PLB Bridge benchmarks are shown in Table 7 for a Virtex-II Pro -7 FPGA using multipass place and route.

Table 7: OPB to PLB Bridge FPGA Performance and Resource Utilization Benchmarks (Virtex-II Pro -7)

	Parameter Values										Device Resources			$f_{MAX}$ (MHZ)
	G1	G4	G9	G10 G11	G18 G19 G20 G21	G22 G23 G24 G25	G26 G27 G28 G29	G30 G31	G35 G36	Slices	SI Flip- flops	4- Input LUTs		
1	1	1	1	x"70000000" x"7FFFFFFF"	0 n/a n/a n/a	0 n/a n/a n/a	0 n/a n/a n/a	x"60000000" x"6FFFFFFF"	n/a	542	697	467	132	
2	1	1	1	x"70000000" x"7000001F"	0 n/a n/a n/a	0 n/a n/a n/a	0 n/a n/a n/a	x"60000000: x"6FFFFFFF"	n/a	556	720	473	128	
3	1	1	1	x"70000000' x'7000001F"	1 n/a n/a n/a	0 n/a n/a n/a	0 n/a n/a n/a	x"60000000" x"6FFFFFFF"	n/a	610	676	644	129	
4	1	0	1	x"70000000" x"7FFFFFFF"	0 n/a n/a n/a	0 n/a n/a n/a	0 n/a n/a n/a	n/a	"1000000 000" "1111111 111"	548	732	475	130	
5	1	1	1	x"70000000" x"7FFFFFFF"	1 n/a n/a n/a	1 n/a n/a n/a	1 n/a n/a n/a	x"60000000" x"6FFFFFFF"	n/a	679	638	886	132	
6	1	1	1	x"70000000" x"7000001F"	0 n/a n/a n/a	0 n/a n/a n/a	1 n/a n/a n/a	x"60000000" x"6FFFFFFF"	n/a	629	702	713	128	

**Table 7: OPB to PLB Bridge FPGA Performance and Resource Utilization Benchmarks (Virtex-II Pro -7)**

7	0	1	1	x"70000000" x"7FFFFFFF"	1 n/a n/a	1 n/a n/a	1 n/a n/a	x"60000000" x"6FFFFFFF"	n/a	690	650	888	130
8	1	1	1	x"70000000" x"7000001F"	1 n/a n/a	0 n/a n/a	1 n/a n/a	x"60000000" x"6FFFFFFF"	n/a	687	656	886	128
9	1	1	2	x"70000000" x"7FFFFFFF" x"80000000" x"FFFFFFFF"	1 1 n/a n/a	1 1 n/a n/a	1 1 n/a n/a	x"60000000" x"6FFFFFFF"	n/a	679	638	889	128
10	1	1	3	x"30000000" x"3FFFFFFF" x"40000000" x"5FFFFFFF" x"E0000000" x"FFFFFFFF"	1 1 1 n/a	1 1 1 n/a	1 1 1 n/a	x"60000000" x"6FFFFFFF"	n/a	681	630	890	129
11	1	1	4	x"00000000" x"3FFFFFFF" x"40000000" x"5FFFFFFF" x"A0000000" x"BFFFFFFF" x"C0000000" x"FFFFFFFF"	1 1 1 1	1 1 1 1	1 1 1 1	x"60000000" x"6FFFFFFF"	n/a	684	638	891	130
12	1	1	1	x"70000000" x"7FFFFFFF"	1 n/a n/a	1 n/a n/a	1 n/a n/a	x"60000000" x"6000001F"	n/a	692	659	892	130

**Table 7: OPB to PLB Bridge FPGA Performance and Resource Utilization Benchmarks (Virtex-II Pro -7)**

13	1	1	1	x"70000000" x"7000001F"	1 n/a n/a n/a	1 n/a n/a n/a	1 n/a n/a n/a	x"60000000" x"6FFFFFFF"	n/a	692	659	892	128
14	1	0	1	x"70000000" x"7FFFFFFF"	1 n/a n/a n/a	1 n/a n/a n/a	1 n/a n/a n/a	n/a	"1000000 000" "1111111 111"	690	678	897	130
15	1	0	1	x"70000000" x"7FFFFFFF"	1 n/a n/a n/a	1 n/a n/a n/a	1 n/a n/a n/a	n/a	"1000000 000" "1000000 011"	694	685	898	130

### Specification Exceptions

This section lists the differences between the Xilinx OPB to PLB Bridge specification (this spec) and the IBM OPB to PLB Bridge specification. Currently, the only noteworthy differences between this specification and IBM specification are listed as follows:

- I/O signal names
- Error register names and definitions
- eXternal Bus Master (XBM) configuration not supported in Xilinx OPB to PLB Bridge

### Reference Documents

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

- *IBM CoreConnect™ 64-bit OPB to PLB Bridge Core User Manual*
- *IBM CoreConnect 64-bit Processor Local Bus: Architecture Specification*
- *IBM CoreConnect 64-bit On-chip Peripheral Bus: Architecture Specification*
- *IBM CoreConnect 32-bit Device Control Register Bus: Architecture Specification*

### Revision History

The following table shows the revision history for this document.

Date	Version	Revision
07/15/03	1.0	Initial Xilinx release.
08/08/03	1.1	Updated resource numbers (F.31)
09/17/03	1.1.1	Correct trademarks
01/27/04	1.1.2	Update copyright and trademarks
06/24/04	1.2	Remove references to any device family except V2P per CR 190407
8/5/04	1.3	Updated trademarks and supported device family listing
9/22/04	1.4	Indicated no support for 32-bit PLB devices
04/24/09	1.5	Replaced references to supported device families and tool name(s) with hyperlinks to PDF files; Updated trademark information.

## Notice of Disclaimer

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

Discontinued IP