PLB to OPB Bridge (v1.01a)

DS403 July 15, 2005 **Product Specification**

Introduction

The Processor Local Bus (PLB) to On-chip Peripheral Bus (OPB) Bridge translates PLB transactions into OPB transactions. It functions as a slave on PLB side and a master on OPB side. It contains a DCR slave interface to provide access to its bus error status registers. The PLB to OPB bridge is necessary in systems where a PLB master device requires access to OPB peripherals. The Xilinx PLB to OPB Bridge design allows customers to tailor the bridge to suit their application by setting certain parameters to enable/disable features. Parameterizable features of the design are discussed in PLB to OPB Bridge Design Parameters. Differences between the IBM PLB to OPB Bridge implementation and the Xilinx PLB to OPB Bridge implementation are also highlighted and explained.

Features

- PLB Slave interface
  - 32-bit or 64-bit PLB (configurable via the C_PLB_DWIDTH design parameter)
  - PLB slave width same as PLB bus width
  - Decodes up to four separate address ranges
  - Programmable lower and upper address boundaries for each range
  - Communicates with 32- or 64-bit PLB masters
  - Non-burst transfers of 1-8 bytes
  - Burst transfers, including word and double-word bursts of fixed or variable lengths, up to depth of burst buffer (16)
  - Limited support for byte, half-word, quad-word and octal-word bursts to maintain PLB compliance
  - Cacheline transactions of 4, 8, and 16 words
  - Support for burst transactions can be eliminated via a design parameter
    - save device resources by only supporting single beat, 4, 8, or 16 word line transfers
  - Supports up to 16 PLB masters (number of PLB masters configurable via a design parameter)
- OPB Master interface with byte enable transfers **Note**:Does not support dynamic bus sizing without additional glue logic
  - Data width configurable via a design parameter

LogiCORE™ Facts		
Core Specifics		
Supported Device Family	Virtex™-II Pro, Virtex-4	
Version of Core	plb2opb_bridge	v1.01a
Resources Used		
	Min	Max
Slices	595	836
LUTs	535	823
FFs	547	812
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	5.1i or later	
Verification	N/A	
Simulation	ModelSim SE/EE 5.6e or later	
Synthesis	XST	
Support		
Support provided by Xilinx, Inc.		

- PLB and OPB clocks can have a 1:1, 2:1, 3:1, and 4:1 synchronous relationship
- Bus Error Address Registers (BEAR) and Bus Error Status Registers (BESR) to report errors
  - DCR Slave interface provides access to BEAR/BESR
  - BEAR, BESR, and DCR interface can be removed from design via a design parameter
- 16-deep posted write buffer and read pre-fetch buffer
- Edge-type interrupt generated when bus error detected
- System compatibility with Xilinx's OPB Master/Slave IPIF architecture when utilized with PLB V34 core version 1.02.a.

PLB and OPB Transaction Behavior

This section describes important information about the behavior of the PLB bridge with respect to the PLB and OPB protocols.

PLB Transaction Qualifiers and Arbitration Signals

- Directly supports 32-/64-bit PLB masters
- Error and busy flags support up to 16 PLB masters
- Accepts only memory transfers (**PLB_type[0:2]** = 000)
  - Other transfer types are ignored
- Ignores **PLB_ordered** qualifier since all transactions are completed in order by design
- Asserts **BGO_rearbitrate** if the bridge is busy with an OPB transaction and a new transaction for the bridge is requested by the PLB arbiter or if an OPB retry is asserted on read transactions.
- Drives **BGO_wait** to zero since it is not utilized
- Introduces an extra clock cycle of latency for address decoding. However, the address decode cycle can be performed during a primary or secondary address cycle.
- Bridge does not support address pipelining with respect to data transfers
- Continuously asserts OPB bus lock (**BGO_busLock**) for as long as the PLB bus remains locked, providing PLB bus lock (**PLB_busLock**) is held after the bridge asserts address acknowledge. Upon receiving a subsequent address acknowledge from the bridge, **BGO_busLock** remains asserted until the data transfer is complete. If a slave asserts a retry, **BGO_busLock** is de-asserted, as required by the OPB specification.

PLB Data Transfers

- Supports non-burst/non-cacheline transfers of 1-8 bytes
- Supports cacheline transfers of 4, 8, or 16 words in linear order
  - Target word first order is not supported.
- Burst transactions are supported as described below if the support for burst transfers has not been parameterized away: (C_NO_PLB_BURST=0)
  - Accepts all burst transfers of bytes (**PLB_size[0:3]** = 1000) but terminates immediately with **BGO_wrBTerm** or **BGO_rdBTerm** after one data transfer
  - Accepts all burst transfers of half words (**PLB_size[0:3]** = 1001) but terminates immediately with **BGO_wrBTerm** or **BGO_rdBTerm** after one data transfer
  - Supports fixed-length burst transfers (**PLB_BE(0:C_PLB_DWIDTH/8 -1)** greater than 0) of words (**PLB_size[0:3]** = 1010) up to 16 words.
  - Supports fixed-length burst transfers of double words (**PLB_size[0:3]** = 1011) up to 8 double words. For double-word bursts greater than the 8, the bridge will terminate the burst with **BGO_wrBTerm** or **BGO_rdBTerm** at the eighth data transfer.
  - Burst transfers of quad words and octal words (**PLB_size[0:3]** = 1100 or 1101) will be treated like double word transfers causing **BGO_wrBTerm** or **BGO_rdBTerm** to be asserted before all data has been transferred. For example, a burst of 3 octal words will be terminated after 3 double words have been transferred.
  - Variable length write bursts (**PLB_BE(0:C_PLB_DWIDTH/8 -1)** = 0) will cause up to 16 words of write data to be queued up. After 16 words of write data have been transferred, the bridge will assert **BGO_wrBTerm** or

BGO_rdBTerm to prevent the master from continuing the variable length burst past the buffer depth. Note that 16 words equal 8 double words so variable length double-word bursts are terminated at the 8 double word.

- Variable length read bursts will cause the bridge to pre-fetch 16 words of data from the OPB. If the master terminates the burst with less than 16 words transferred, the OPB master will terminate any reads that have not been performed. (Any OPB read transaction in progress is allowed to complete). Note that the pipelined architecture of the bridge and the use of pre-fetching implies that the OPB master reads data in advance of knowing when the PLB master stops the burst. This may cause unnecessary read cycles over OPB. PLB masters should only use variable length read bursts when the data source is pre-fetchable to prevent the loss of data. Otherwise, PLB masters should use fixed length bursts so that only the desired data is read.
- Fixed and variable length bursts are terminated at 1 KB address boundaries if the guarded attribute is set (**PLB_guarded** = 1). Pre-fetched read data does not cross the guarded 1K address boundary during a variable length read burst.

- Burst transactions are not supported if the parameter C_NO_PLB_BURST = 1. In this case, the PLB2OPB Bridge will not respond to any burst-type transactions (PLB_size[0:3] = "1xxx") , i.e, BGO_addrAck will not be asserted. This will cause a PLB time-out and if the PLB error and status registers in the PLB have not been previously locked, the time-out information will be contained in these registers.

OPB Data Transfers

- Master interface only supports 32-bit, byte enable slaves
- OPB byte enable transfer (**M_beXfer**) port is not present because it is implicitly asserted with **BGO_select**
- OPB byte enable and full word acknowledge (**OPB_beAck**/**OPB_fwAck**) are not present because a slave would implicitly assert these signals with **OPB_xferAck**.
- Does not perform dynamic bus sizing operations for byte or half word OPB slaves
- OPB half-word transfer/acknowledge (**M_hwXfer**/**OPB_hwAck**) and full word transfer/acknowledge (**M_fwXfer**/**OPB_fwAck**) signals are not present
- Data is not mirrored over inactive byte lanes. If a byte enable bit is zero, the corresponding data byte lane has an undefined value
- Master output signals do not need AND-OR gating logic in the OPB bus since they are gated internally to improve bus timing. OR-gates are sufficient for the OPB bus logic connection
- Master abort cycles are not generated
- Interface Logic is available to allow legacy dynamic bus sizing devices to communicate with the bridge. However, OPB peripherals should be designed to directly support byte enable transfers for highest performance and lowest resource utilization.
- If **OPB_retry** is asserted instead of **OPB_xferAck,** the transaction will be retried until successful. The OPB bus will be released as required by the OPB specification for one clock after each receipt of **OPB_retry**.
- If the OPB transaction is terminated with **OPB_timeout** or **OPB_errAck**, the **Bus_Error_Det** interrupt signal is asserted and the Bus Error Status Registers and the Bus Error Address Registers are updated. **PLB to OPB Bridge Register Descriptions**

PLB to OPB Bridge Design Parameters

To allow the user to obtain a PLB to OPB Bridge that is uniquely tailored for a specific system, certain features can be parameterized in the Xilinx PLB to OPB Bridge design, thereby producing a design that utilizes only the resources required by the system and that will run at the best possible performance. The features that can be parameterized in the Xilinx PLB to OPB Bridge are shown in Table 1.

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

Grouping/Number		Feature / Description	Parameter Name	Allowable Values	Default Value	VHDL Type
Bridge Features	G1	No support for burst transactions(1)	C_NO_PLB_BURST	1 = Support only single beat, 4, 8, and 16 word line transfers 0 = Support all CoreConnect transactions	0	integer
	G2	Include DCR slave interface	C_DCR_INTFCE	1 = Include DCR slave interface 0 = DCR slave interface not included	1	integer
	G3	Target device family	C_FAMILY	spartan2, spartan2e, virtex, virtexe, virtex2, virtex2p	virtex2p	string
PLB Slave Interface	G4	Number of PLB address ranges	C_NUM_ADDR_RNG	1 - 4	1	integer
	G5	PLB Base Address for address range 1	C_RNG0_BASEADDR	Valid Address Range(2)	None(4)	std_logic_vector
	G6	PLB High Address for address range 1	C_RNG0_HIGHADDR	Address range must be a power of 2(2)	None(4)	std_logic_vector
	G7	PLB Base Address for address range 2	C_RNG1_BASEADDR	Valid Address Range(2)	None(4)	std_logic_vector
	G8	PLB High Address for address range 2	C_RNG1_HIGHADDR	Address range must be a power of 2(2)	None(4)	std_logic_vector
	G9	PLB Base Address for address range 3	C_RNG2_BASEADDR	Valid Address Range(2)	None(4)	std_logic_vector
	G10	PLB High Address for address range 3	C_RNG2_HIGHADDR	Address range must be a power of 2(2)	None(4)	std_logic_vector
	G11	PLB Base Address for address range 4	C_RNG3_BASEADDR	Valid Address Range(2)	None(4)	std_logic_vector
	G12	PLB High Address for address range 4	C_RNG3_HIGHADDR	Address range must be a power of 2(2)	None(4)	std_logic_vector
	G13	PLB Address Bus Width	C_PLB_AWIDTH	32(5)	32	integer
	G14	PLB Data Bus Width	C_PLB_DWIDTH	64	64	integer
	G15	Number of PLB Masters	C_PLB_NUM_MASTERS	1 - 16	4	integer
OPB Master Interface	G16	OPB Address Bus Width	C_OPB_AWIDTH	32(5)	32	integer
	G17	OPB Data Bus Width	C_OPB_DWIDTH	32	32	integer

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

Grouping/Number		Feature / Description	Parameter Name	Allowable Values	Default Value	VHDL Type
DCR Interface	G18	DCR Base Address	C_DCR_BASEADDR	Valid DCR address(3)	None(4)	std_logic_vector
	G19	DCR High Address	C_DCR_HIGHADDR	Valid DCR address(3)	None(4)	std_logic_vector
	G20	DCR Address Bus Width	C_DCR_AWIDTH	10	10	integer
	G21	DCR Data Bus Width	C_DCR_DWIDTH	32	32	integer
Interrupts	G22	Active interrupt state(6)	C_IRQ_ACTIVE	'0' = interrupt request will be driven as a falling edge '1' = interrupt request will be driven as a rising edge	'1'	std_logic
Deadlock Prevention	G23	Number of clocks to wait for OPB Grant before instruction BGI to abort transaction	C_BGI_TRANSABORT_CNT	1 - 31	31	integer
Auto-calculated para-meters(7)	G24	Number of bits required to encode the number of PLB Masters	C_PLB_MID_WIDTH	1 - log2(C_NUM_MASTERS)	2	integer
Reserved Para-meters(8)	G25	Reserved	C_CLK_ASYNC	1	1	integer
	G26	Reserved	C_HIGH_SPEED	1	1	integer
	G27	Reserved	C_INCLUDE_BGI_TRANSABORT	1	1	integer

Notes:

1. Only support single beat, 4, 8, and 16 word line transfers. Setting this parameter to 1 eliminates all support in the bridge for burst transactions. If a burst transaction is received when this parameter is set to 1, BGO_addrAck will not be asserted, forcing a PLB time-out.
2. Four sets of address ranges can be specified for the bridge. The number of address ranges needed is set in the parameter C_NUM_ADDR_RNG. The range specified by the various base addresses and corresponding high addresses must comprise a complete, contiguous power of two range such that range = 2^n, and the n least significant bits of the base address must be zero. If an address range needs to support 16 word cacheline transactions, the base address for this address range must be aligned to a 64-byte address. If an address range contains guarded memory, the high address for this address range must be aligned to a 1K byte address.
3. The range specified by C_DCR_BASEADDR and C_DCR_HIGHADDR must comprise a complete, contiguous power of two range such that range = 2^n, and the n least significant bits of C_DCR_BASEADDR must be zero. To allow for the 8 DCR registers within the PLB to OPB Bridge, n must be at least 3.
4. No default value will be specified for the base addresses or high addresses to insure that the actual value is set, i.e. if the value is not set, a compiler error will be generated.
5. The width of the PLB address bus and the OPB address bus should be equal.
6. The interrupt request output (Bus_Error_Det, P55) is generated as an edge type interrupt. A specific interrupt acknowledge response is not required.
7. These parameters are automatically calculated by the system generation tool and are not input by you.
8. These parameters are reserved and are not user modifiable.

Allowable Parameter Combinations

The address range specified by the various PLB base addresses and high addresses must comprise a complete, contiguous power of two range such that each range = 2^n, and the n least significant bits of the base address for that range must be zero.

If an address range needs to support 16 word cacheline transactions, the base address for this address range must be aligned to a 64-byte address. If an address range contains guarded memory, the high address for this address range must be aligned to a 1K byte address.

The address range specified by C_DCR_BASEADDR and C_DCR_HIGHADDR must also comprise a complete, contiguous power of two range such that range = 2^n, and the n least significant bits of C_DCR_BASEADDR must be zero.

To allow for the registers in the PLB to OPB Bridge design, this range must be at least 8; therefore n must be at least 3. This means that at a minimum, the three least significant bits of C_DCR_BASEADDR must be 0.

The base address and high address parameters determine the number of most significant address bits used to decode the address space. These parameters allow you to trade-off address space resolution with size and speed of the bridge.

The width of the PLB address bus and width of the OPB address bus must be equal.

Note that 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 address that will fit into a DCR data word.

Some parameters can cause other parameters to be irrelevant. See Table 3 for information on the relationship between design parameters.

PLB to OPB Bridge I/O Signals

The I/O signals for the PLB to OPB Bridge are listed in Table 2. The interfaces referenced in Table 2 are shown in Figure 9 in the PLB to OPB Bridge block diagram.

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

Grouping		Signal Name	Interface	I/O	Initial State	Description
PLB Slave Signals	P1	PLB_PAValid	PLB	I		PLB primary address valid indicator
	P2	PLB_busLock	PLB	I		PLB bus lock
	P3	PLB_masterID(0:C_PLB_MID_WIDTH-1)	PLB	I		PLB current master identifier
	P4	PLB_RNW	PLB	I		PLB read not write
	P5	PLB_BE(0:C_PLB_DWIDTH/8 -1)	PLB	I		PLB byte enables
	P6	PLB_size[0:3]	PLB	I		PLB transfer size
	P7	PLB_type[0:2]	PLB	I		PLB transfer type
	P8	PLB_MSize[0:1]	PLB	I		PLB master data bus size
	P9	PLB_compress	PLB	I		PLB compressed data transfer indicator
	P10	PLB_guarded	PLB	I		PLB guarded transfer indicator
	P11	PLB_ordered	PLB	I		PLB synchronize transfer indicator
	P12	PLB_lockErr	PLB	I		PLB lock error indicator
	P13	PLB_abort	PLB	I		PLB abort bus request indicator
	P14	PLB_ABus(0:C_PLB_AWIDTH-1)	PLB	I		PLB address bus
	P15	BGO_addrAck	PLB	O	0	Slave address acknowledge

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

Grouping		Signal Name	Interface	I/O	Initial State	Description
	P16	BGO_wait	PLB	O	0	Slave wait indicator
	P17	BGO_SSize[0:1]	PLB	O	0	Slave data bus size
	P18	BGO_rearbitrate	PLB	O	0	Slave rearbitrate bus indicator
	P19	BGO_MBusy(0:C_PLB_NUM_MASTERS-1)	PLB	O	0	Slave busy indicator
	P20	BGO_MErr(0:C_PLB_NUM_MASTERS-1)	PLB	O	0	Slave error indicator
	P21	PLB_SAValid	PLB	I		PLB secondary address valid indicator
	P22	PLB_rdPrim	PLB	I		PLB secondary to primary read request indicator
	P23	PLB_wrPrim	PLB	I		PLB secondary to primary write request indicator
	P24	PLB_wrDBus(0:C_PLB_DWIDTH-1)	PLB	I		PLB write data bus
	P25	PLB_wrBurst	PLB	I		PLB burst write transfer indicator
	P26	BGO_wrDAck	PLB	O	0	Slave read data acknowledge
	P27	BGO_wrComp	PLB	O	0	Slave write transfer complete indicator
	P28	BGO_wrBTerm	PLB	O	0	Slave terminate write burst transfer
	P29	PLB_rdBurst	PLB	I		PLB burst read transfer indicator
	P30	BGO_rdDBus(0:C_PLB_DWIDTH-1)	PLB	O		Slave read data bus
	P31	BGO_rdWdAddr(0:3)	PLB	O		Slave read word address
	P32	BGO_rdDAck	PLB	O		Slave read data acknowledge
	P33	BGO_rdComp	PLB	O		Slave read transfer complete indicator
	P34	BGO_rdBTerm	PLB	O		Slave terminate read burst transfer
OPB Master Signals	P35	OPB_DBus(0:C_OPB_DWIDTH-1)	OPB	I		OPB data bus
	P36	OPB_errAck	OPB	I		OPB error acknowledge
	P37	OPB_MnGrant	OPB	I		OPB master bus grant
	P38	OPB_retry	OPB	I		OPB retry
	P39	OPB_timeout	OPB	I		OPB timeout error
	P40	OPB_xferAck	OPB	I		OPB transfer acknowledge
	P41	BGO_Abus(0:C_OPB_AWIDTH-1)	OPB	O	0	Master address bus
	P42	BGO_BE(0:C_OPB_DWIDTH/8 -1)	OPB	O	0	Master byte enables
	P43	BGO_busLock	OPB	O	0	Master bus arbitration lock
	P44	BGO_DBus(0:C_OPB_DWIDTH-1)	OPB	O	0	Master data bus
	P45	BGO_request	OPB	O	0	Master bus request
	P46	BGO_RNW	OPB	O	0	Master read not write
	P47	BGO_select	OPB	O	0	Master select

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

Grouping		Signal Name	Interface	I/O	Initial State	Description
	P48	BGO_seqAddr	OPB	O	0	Master sequential address
DCR Signals	P49	DCR_ABus(0:C_DCR_AWIDTH-1)	DCR	I		DCR address bus
	P50	DCR_DBus(0:C_DCR_DWIDTH-1)	DCR	I		DCR data bus in
	P51	DCR_Read	DCR	I		DCR read
	P52	DCR_Write	DCR	I		DCR write
	P53	BGO_dcrAck	DCR	O	0	DCR acknowledge
	P54	BGO_dcrDBus(0:C_DCR_DWIDTH-1)	DCR	O	0	DCR data bus out
Interrupt	P55	Bus_Error_Det	System	O	0	Bus error interrupt
System	P56	OPB_Clk	System	I		OPB clock
	P57	PLB_Clk	System	I		PLB System clock
	P58	OPB_Rst	System	I		OPB Bus Reset (active high)(1)
	P59	PLB_Rst	System	I		PLB Bus Reset(2)
	P60	BGI_Trans_Abort	System	O	0	Signals BGI to abort current transaction
	P61	PLB2OPB_rearb	PLB	O	0	Signals rearbitration on read is asserted by plb2opb bridge

Notes:
1. The bridge is reset using the OPB Bus reset. This reset needs to be active high for at least 2 OPB clock periods to insure that the portions of the bridge that are clocked by the PLB clock are correctly reset.
2. The PLB bus reset signal is not used. It is part of the PLB2OPB Bridge I/O to provide a consistent PLB slave signal set.

Parameter-Port Dependencies

The width of many of the PLB to OPB Bridge signals depends on the number of PLB masters in the system and the width of the various data and address busses. In addition, when certain features are parameterized away, the related input signals are unconnected and the related output signals are set to a constant values. The dependencies between the PLB to OPB Bridge design parameters and I/O signals are shown in Table 3.

Table 3: **Parameter-Port Dependencies**

		Name	Affects	Depends	Relationship Description
Design Parameters	G1	C_NO_PLB_BURST			
	G2	C_DCR_INTFCE	P49 - P54		If C_DCR_INTFCE=0 then the I/O signals associated with the DCR interface are unconnected. BEAR and BESR registers are not accessible. Interrupts are enabled, however, so you can connect the Bus_Error_Det signal as desired.
	G3	C_FAMILY			
	G4	C_NUM_ADDR_RNG			Number of PLB address ranges to be decoded

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

	Name	Affects	Depends	Relationship Description
G5	C_RNG0_BASEADDR	G6	G6,G13	Range specified by C_RNG0_BASEADDR and C_RNG0_HIGHADDR must be a power of 2. Width of C_RNG0_BASEADDR is determined by C_PLB_AWIDTH.
G6	C_RNG0_HIGHADDR	G5	G5,G13	Range specified by C_RNG0_BASEADDR and C_RNG0_HIGHADDR must be a power of 2. Width of C_RNG0_HIGHADDR is determined by C_PLB_AWIDTH.
G7	C_RNG1_BASEADDR	G8	G4,G8,G13	Range specified by C_RNG1_BASEADDR and C_RNG1_HIGHADDR must be a power of 2. Width of C_RNG1_BASEADDR is determined by C_PLB_AWIDTH. If C_NUM_ADDR_RNG < 2, this address range is unused.
G8	C_RNG1_HIGHADDR	G7	G4,G7,G13	Range specified by C_RNG1_BASEADDR and C_RNG1_HIGHADDR must be a power of 2. Width of C_RNG1_HIGHADDR is determined by C_PLB_AWIDTH. If C_NUM_ADDR_RNG < 2, this address range is unused.
G9	C_RNG2_BASEADDR	G10	G4,G10,G13	Range specified by C_RNG2_BASEADDR and C_RNG2_HIGHADDR must be a power of 2. Width of C_RNG2_BASEADDR is determined by C_PLB_AWIDTH. If C_NUM_ADDR_RNG < 3, this address range is unused.
G10	C_RNG2_HIGHADDR	G9	G4,G9,G13	Range specified by C_RNG2_BASEADDR and C_RNG2_HIGHADDR must be a power of 2. Width of C_RNG2_HIGHADDR is determined by C_PLB_AWIDTH. If C_NUM_ADDR_RNG < 3, this address range is unused.

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

		Name	Affects	Depends	Relationship Description
	G11	C_RNG3_BASEADDR	G12	G4,G12, G13	Range specified by C_RNG3_BASEADDR and C_RNG3_HIGHADDR must be a power of 2. Width of C_RNG3_BASEADDR is determined by C_PLB_AWIDTH. If C_NUM_ADDR_RNG < 4, this address range is unused.
	G12	C_RNG3_HIGHADDR	G11	G4,G11, G13	Range specified by C_RNG3_BASEADDR and C_RNG3_HIGHADDR must be a power of 2. Width of C_RNG3_HIGHADDR is determined by C_PLB_AWIDTH. If C_NUM_ADDR_RNG < 4, this address range is unused.
	G13	C_PLB_AWIDTH	G5 - G12 P14	G16	The width of the PLB address bus and the OPB address bus must be equal.
	G14	C_PLB_DWIDTH	P5, P24, P30		
	G15	C_PLB_NUM_MASTERS	G24, P19, P20		
	G16	C_OPB_AWIDTH	P41		The width of the PLB address bus and the OPB address bus must be equal.
	G17	C_OPB_DWIDTH	P35, P42, P44	G13	
	G18	C_DCR_BASEADDR		G2, G20	Unconnected if C_DCR_INTFCE=0. Width of C_DCR_BASEADDR is determined by C_DCR_AWIDTH.
	G19	C_DCR_HIGHADDR		G2	Unconnected if C_DCR_INTFCE=0.
	G20	C_DCR_AWIDTH		G2	Unconnected if C_DCR_INTFCE=0.
	G21	C_DCR_DWIDTH		G2	Unconnected if C_DCR_INTFCE=0.
	G22	C_IRQ_ACTIVE	P55		
	G23	C_BGI_TRANSABORT_CNT			
	G24	C_PLB_MID_WIDTH	P3	G15	
I/O Signals	P1	PLB_PAValid			
	P2	PLB_busLock			
	P3	PLB_masterID(0:C_PLB_MID_WIDTH-1)		G24	Width varies with the number of bits required to encode the number of PLB masters.

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

	Name	Affects	Depends	Relationship Description
P4	PLB_RNW			
P5	PLB_BE(0:C_PLB_DWIDTH/8 -1)		G14	Width varies with the width of the PLB data bus.
P6	PLB_size[0:3]			
P7	PLB_type[0:2]			
P8	PLB_MSize[0:1]			
P9	PLB_compress			
P10	PLB_guarded			
P11	PLB_ordered			
P12	PLB_lockErr			
P13	PLB_abort			
P14	PLB_ABus(0:C_PLB_AWIDTH-1)		G13	Width varies with the width of the PLB address bus.
P15	BGO_addrAck			
P16	BGO_wait			
P17	BGO_SSize[0:1]			
P18	BGO_rearbitrate			
P19	BGO_MBusy(0:C_PLB_NUM_MASTERS-1)		G15	Width varies with the number of PLB masters.
P20	BGO_MErr(0:C_PLB_NUM_MASTERS-1)		G15	Width varies with the number of PLB masters.
P21	PLB_SAValid			
P22	PLB_rdPrim			
P23	PLB_wrPrim			
P24	PLB_wrDBus(0:C_PLB_DWIDTH-1)		G14	Width varies with the width of the PLB data bus.
P25	PLB_wrBurst			
P26	BGO_wrDAck			
P27	BGO_wrComp			
P28	BGO_wrBTerm			
P29	PLB_rdBurst			
P30	BGO_rdDBus(0:C_PLB_DWIDTH-1)		G14	Width varies with the width of the PLB data bus.
P31	BGO_rdWdAddr(0:3)			
P32	BGO_rdDAck			
P33	BGO_rdComp			
P34	BGO_rdBTerm			

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

	Name	Affects	Depends	Relationship Description
P35	OPB_DBus(0:C_OPB_DWIDTH-1)		G17	Width varies with the width of the OPB data bus.
P36	OPB_errAck			
P37	OPB_MnGrant			
P38	OPB_retry			
P39	OPB_timeout			
P40	OPB_xferAck			
P41	BGO_Abus(0:C_OPB_AWIDTH-1)		G16	Width varies with the width of the OPB address bus.
P42	BGO_BE(0:C_OPB_DWIDTH/8 -1)		G17	Width varies with the width of the OPB data bus.
P43	BGO_busLock			
P44	BGO_DBus(0:C_OPB_DWIDTH-1)		G17	Width varies with the width of the OPB data bus.
P45	BGO_request			
P46	BGO_RNW			
P47	BGO_select			
P48	BGO_seqAddr			
P49	DCR_ABus(0:C_DCR_AWIDTH-1)		G2, G19	This input is unconnected if C_DCR_INTFCE = 0. Width varies with the width of DCR address bus.
P50	DCR_DBus(0:C_DCR_DWIDTH-1)		G2, G20	This input is unconnected if C_DCR_INTFCE = 0. Width varies with the width of DCR data bus.
P51	DCR_Read		G2	This input is unconnected if C_DCR_INTFCE=0.
P52	DCR_Write		G2	This input is unconnected if C_DCR_INTFCE=0.
P53	BGO_dcrAck		G2	This output is grounded if C_DCR_INTFCE=0.
P54	BGO_dcrDBus(0:C_DCR_DWIDT H-1)		G2, G20	This output is grounded if C_DCR_INTFCE=0. Width varies with the width of DCR data bus.
P55	Bus_Error_Det		G2, G22	This output is grounded if C_DCR_INTFCE=0 because interrupts are disabled in this case. C_IRQ_ACTIVE sets the active level of this output when interrupts are enabled.
P56	OPB_Clk			
P57	PLB_Clk			

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

	Name	Affects	Depends	Relationship Description
P58	OPB_Rst			
P59	PLB_Rst			
P60	BGI_Trans_Abort			

PLB to OPB Bridge Register Descriptions

The PLB to OPB Bridge contains eight DCR-accessible registers to provide error address and status information if the design has been parameterized to contain the Bus Error Address Registers (BEAR) and Bus Error Status Registers (BESR) as shown in Table 4. The base address for these registers is set in the parameter C_DCR_BASEADDR.

Table 4: **PLB to OPB Bridge DCR Registers**

Register Name	Description	DCR Address	Access
BESR_MERR_DETECT	Master Error Detect Bits	C_DCR_BASEADDR + 0x00	Read/Write
BESR_MDRIVE_BEAR	Master Driving BEAR	C_DCR_BASEADDR + 0x01	Read
BESR_RNW_ERR	Read/Write Error	C_DCR_BASEADDR + 0x02	Read
BESR_ERR_TYPE	Error Type	C_DCR_BASEADDR + 0x03	Read
BESR_LCK_ERR	Lock Error Bit	C_DCR_BASEADDR + 0x04	Read
BEAR_ADDR	Bus Error Address	C_DCR_BASEADDR + 0x05	Read
BEAR_BYTE_EN	Bus Error Byte Enables	C_DCR_BASEADDR + 0x06	Read
BGO_CTRL_REG	Bridge Out Control Register	C_DCR_BASEADDR + 0x07	Read/Write

Bus Error Status Registers

There are five BESR registers that provide error information - was an error detected, which Master's address is in the BEAR, was the error due to a read or write transaction, was the error a time-out or an error acknowledge, and did the master lock the error condition. If a read of the BESR_MERR_DETECT register returns all zeros, then no masters detected any errors and no further reads are necessary.

BESR_MERR_DETECT: Master Error Detect Bits

This register contains the error detect bit for each master. The bit location corresponds to the PLB Master. For example, if PLB Master 0 has detected an error, then bit 0 will be set. Writing a '1' to a bit in this register clears this bit and the corresponding bit in the other BESRs (BESR_MDRIVE_BEAR, BESR_RNW_ERR, BESR_ERR_TYPE, and BESR_LCK_ERR).

If a particular master detected an error and had locked the BEARs, writing a '1' to the corresponding bit in this register would clear and unlock the master's error fields and unlock the BEARs. The bits in this register are reset when a '1' has been written to the register, OPB_Rst has been asserted, or a '1' has been written to the Software Reset bit in the BGO_CTRL_REG. Figure 1 shows the bit definitions of this register when the number of PLB masters is 8 and the width of the DCR data bus is 32.

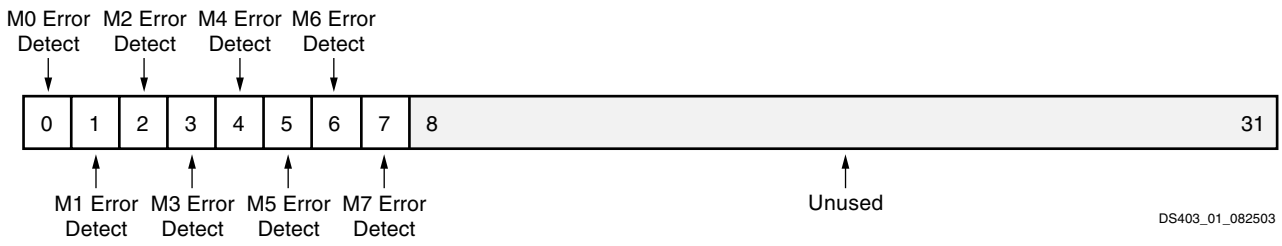

Figure 1: **BESR_MERR_DETECT (C_PLB_NUM_MASTERS=8, C_DCR_DWIDTH=32)**

The bit definitions for BESR_MERR_DETECT are shown in Table 5. The bits is this register are reset by writing a '1' to the bit.

Table 5: **PLB to OPB Bridge BESR_MERR_DETECT Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0 to C_PLB_NUM_MASTERS-1	Master Error Detect	Read/Write	'0'	Master Error Detect. Read: Error detect bit for PLB Masters 0 to C_PLB_NUM_MASTERS-1 respectively. • '1' - error detected • '0' - no error detected Write: Clear error bit for PLB Masters 0 to C_PLB_NUM_MASTERS -1 respectively. • '1' - clear error • '0' - do not clear error
C_PLB_NUM_MASTERS to C_DCR_DWIDTH-1	Unused			

BESR_MDRIVE_BEAR: Master Driving BEAR

This register indicates which PLB Master is driving the BEAR. Each bit location in this register corresponds to a PLB Master.

For example, if PLB Master 0 is driving the BEAR, then bit 0 will be set. Only one master can drive the BEAR, therefore, only one bit will be set in this register. Writing to this register has no effect.

The bits in this register are reset when a '1' is written to the corresponding bits in BESR_MERR_DETECT, OPB_Rst has been asserted, or a '1' has been written to the Software Reset bit in the BGO_CTRL_REG. Figure 2 shows the bit definitions of this register when the number of PLB masters is 8 and the width of the DCR data bus is 32.

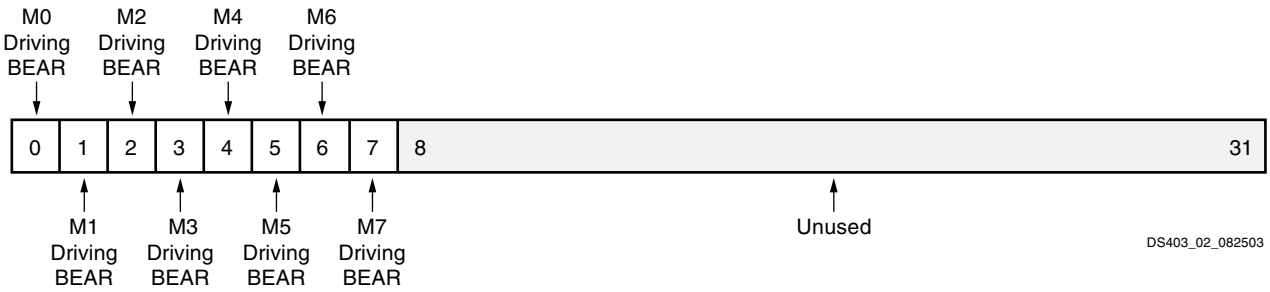

Figure 2: **BESR_MDRIVE_BEAR (C_PLB_NUM_MASTERS=8, C_DCR_DWIDTH=32)**

The bit definitions for BESR_MDRIVE_BEAR are shown in Table 6.

Table 6: **PLB to OPB Bridge BESR_MDRIVE_BEAR Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0 to C_PLB_NUM_MASTERS-1	Master Driving BEAR	Read	'0'	Master Driving BEAR. Read: BEAR bit for PLB Masters 0 to C_PLB_NUM_MASTERS-1 respectively. • '1' - master is driving BEAR • '0' - master is not driving BEAR Write: No effect.
C_PLB_NUM_MASTERS to C_DCR_DWIDTH-1	Unused			

BESR_RNW_ERR: Master Read/Write Bits

This register indicates the read/write condition that caused the error for each PLB Master. Each bit location in this register corresponds to a PLB Master.

For example, if PLB Master 0 detected an error during a read operation, bit 0 would be set. If PLB Master 1 detected an error during a write operation, bit 1 would be reset. Writing to this register has no effect.

The bits in this register are reset when a '1' is written to the corresponding bits in BESR_MERR_DETECT, OPB_Rst has been asserted, or a '1' has been written to the Software Reset bit in the BGO_CTRL_REG. Figure 3 shows the bit definitions of this register when the number of PLB masters is 8 and the width of the DCR data bus is 32.

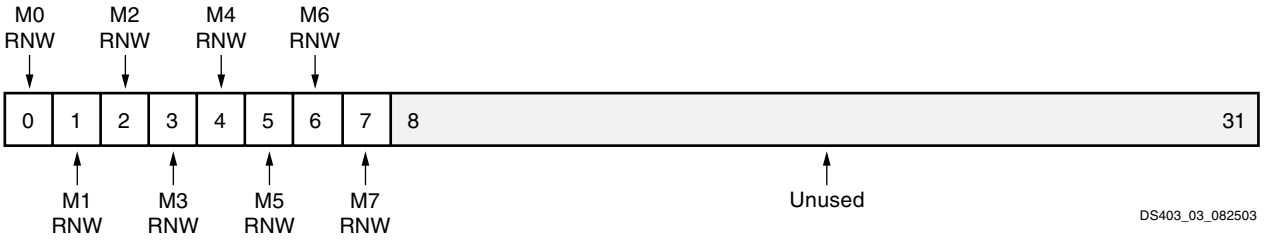

Figure 3: **BESR_RNW_ERR (C_PLB_NUM_MASTERS=8, C_DCR_DWIDTH=32)**

The bit definitions for BESR_RNW_ERR are shown in Table 7.

Table 7: **PLB to OPB Bridge BESR_RNW_ERR Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0 to C_PLB_NUM_MASTERS-1	Master Read Not Write	Read	'0'	Master Read Not Write. Read: RNW status for each master • '1' - error was in response to a read • '0' - error was in response to a write Write: No effect.
C_PLB_NUM_MASTERS to C_DCR_DWIDTH-1	Unused			

BESR_ERR_TYPE: Master Error Type Bits

This register indicates the condition (time-out or error acknowledge) that caused the error for each PLB Master. Each bit location in this register corresponds to a PLB Master.

For example, if the error PLB Master 0 detected was due to a time-out, bit 0 would be set. If the error detected by PLB Master 1 was due to an error acknowledge response from the slave, bit 1 would be reset. Writing to this register has no effect.

The bits in this register are reset when a '1' is written to the corresponding bits in BESR_MERR_DETECT, OPB_Rst has been asserted, or a '1' has been written to the Software Reset bit in the BGO_CTRL_REG. Figure 4 shows the bit definitions of this register when the number of PLB masters is 8 and the width of the DCR data bus is 32.

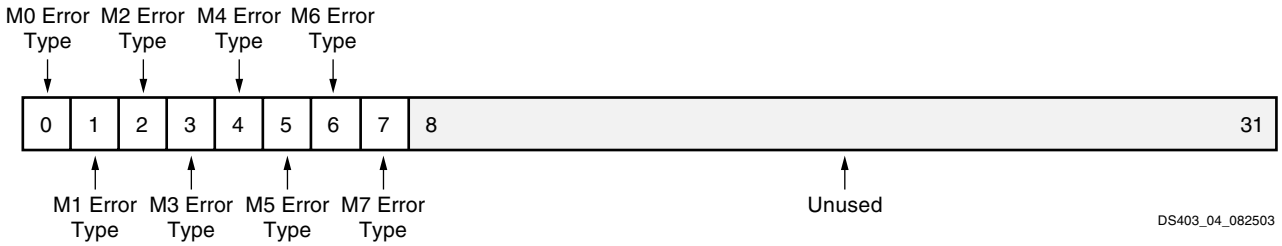

Figure 4: **BESR_ERR_TYPE (C_PLB_NUM_MASTERS=8, C_DCR_DWIDTH=32)**

The bit definitions for BESR_ERR_TYPE are shown in Table 8.

Table 8: **PLB to OPB Bridge BESR_ERR_TYPE Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0 to C_PLB_NUM_MASTERS-1	Master Error Type	Read	'0'	Master Read Not Write. Read: Error type for each master • '1' - error was due to OPB Timeout • '0' - error was due to OPB Error Acknowledge Write: No effect.
C_PLB_NUM_MASTERS to C_DCR_DWIDTH-1	Unused			

BESR_LCK_ERR: Master Lock Error Bits

This register indicates whether each PLB Master has locked their error bits. Each bit location in this register corresponds to a PLB Master.

Setting the Master's lock error bit means that the master's error fields are locked, i.e., subsequent errors cannot overwrite master's error fields until error is cleared. If the Master's lock error bit is reset, the master's error fields are not locked and subsequent errors will overwrite the master's error fields.

Writing to this register has no effect. The bits in this register are reset when a '1' is written to the corresponding bits in BESR_MERR_DETECT, OPB_Rst has been asserted, or a '1' has been written to the Software Reset bit in the BGO_CTRL_REG. Figure 5 shows the bit definitions of this register when the number of PLB masters is 8 and the width of the DCR data bus is 32.

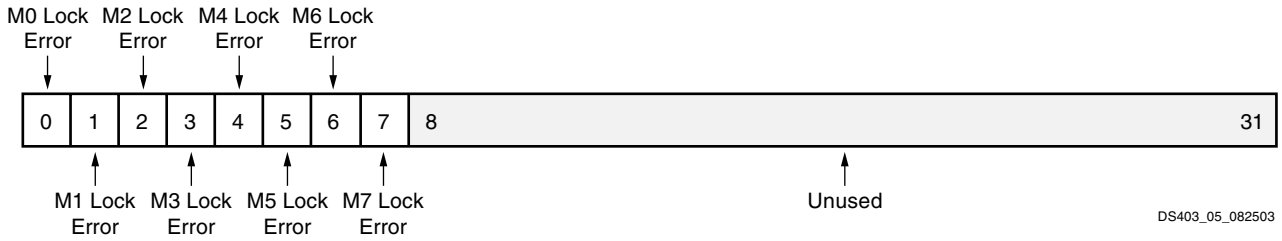

Figure 5: **BESR_LCK_ERR (C_PLB_NUM_MASTERS=8, C_DCR_DWIDTH=32)**

The bit definitions for BESR_LCK_ERR are shown in Table 9.

Table 9: **PLB to OPB Bridge BESR_LCK_ERR Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0 to C_PLB_NUM_MASTERS-1	Master Lock Error	Read	'0'	Master Lock Error. Read: Lock error bit for each master • '1' -error fields are locked (subsequent errors cannot overwrite master's error fields until error is cleared) • '0' - error fields are not locked (subsequent errors can overwrite master's error fields) Write: No effect.
C_PLB_NUM_MASTERS to C_DCR_DWIDTH-1	Unused			

Bus Error Address Registers

There are two BEAR registers - one contains the OPB address of the transaction that caused the error and the other contains the value of the OPB byte enables during the transaction that caused the error.

BEAR_ADDR: Bus Error Address

This register contains the OPB address of the transaction that caused the error as shown in Figure 6. Note that the width of the OPB address bus must be <= the width of the DCR data bus for the entire OPB address to be stored.

Otherwise, only the most significant bits of the OPB address are stored. This register is cleared when OPB_Rst is asserted or a '1' has been written to the Software Reset bit in the BGO_CTRL_REG.

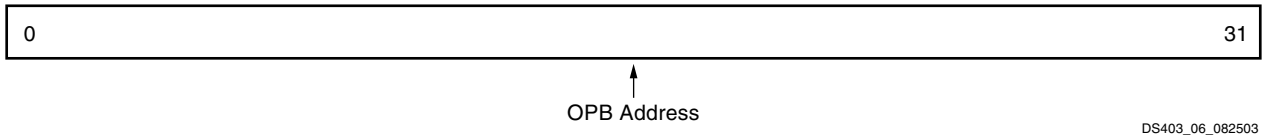

Figure 6: **BEAR_ADDR (C_OPB_AWIDTH=32, C_DCR_DWIDTH=32)**

The bit definitions for BEAR_ADDR are shown in Table 10.

Table 10: **PLB to OPB Bridge BEAR_ADDR Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0 to C_OPB_AWIDTH-1	Bus Error Address	Read	'0'	Bus Error Address. Read: OPB address where error occurred Write: No effect.
C_OPB_AWIDTH to C_DCR_DWIDTH-1	Unused			

BEAR_BYTE_EN: Bus Error Byte Enables

This register contains the values of the OPB byte enables during the transaction that caused the error as shown in Figure 7. The width of the OPB byte enable bus is the width of the OPB data bus divided by 8. Therefore, if the OPB data bus is 64 bits wide, there are 8 byte enables. This register is cleared when OPB_Rst is asserted or a '1' has been written to the Software Reset bit in the BGO_CTRL_REG.

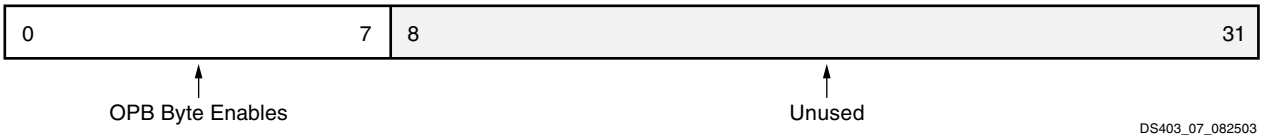

Figure 7: **BEAR_BYTE_EN (C_OPB_DWIDTH=64, C_DCR_DWIDTH=32)**

The bit definitions for BEAR_BYTE_EN are shown in Table 11.

Table 11: **PLB to OPB Bridge BEAR_BYTE_EN Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0 to C_OPB_DWIDTH/8-1	Bus Error Address	Read	'0'	Bus Error Address. Read: OPB byte enable value when error occurred Write: No effect.
C_OPB_DWIDTH/8 to C_DCR_DWIDTH-1	Unused			

Bridge Out Control Register

There is one Bridge Out Control register that enables or disables the interrupt request output from the PLB to OPB bridge and provides a software reset.

BGO_CTRL_REG: Bridge Out Control Register

This register contains one bit that enables or disables the interrupt request and another bit used to reset the PLB to OPB bridge as shown in Figure 8.

Note that the default state of the control register is to have interrupts enabled, therefore if the PLB to OPB bridge is parameterized to not have a DCR interface (C_DCR_INTFCE = 0) then interrupts are still enabled.

Also note that when the BGO reset bit is asserted, ALL registers and flip-flops within the PLB to OPB bridge including all DCR registers are reset. This reset occurs independent of the current PLB and OPB transaction states, therefore, this reset should be used carefully. This register is reset to the default state whenever OPB_Rst is asserted or a '1' is written to the Software Reset bit.

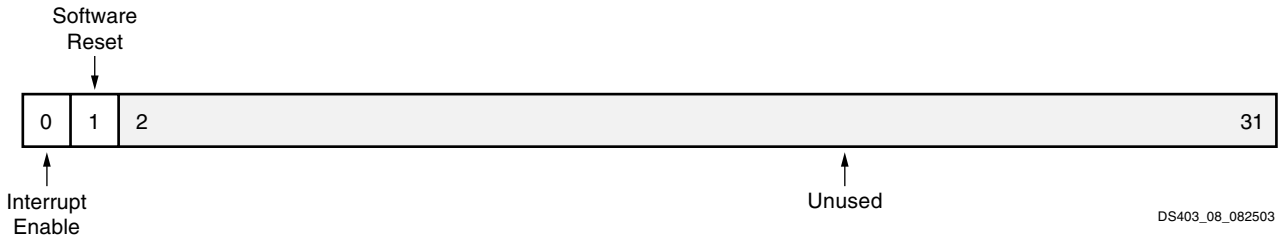

Figure 8: **BGO_CTRL_REG (C_DCR_DWIDTH=32)**

The bit definitions for BGO_CTRL_REG are shown in Table 12.

Table 12: **PLB to OPB Bridge BGO_CTRL_REG Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0	Interrupt Enable	Read/Write	'1'	Interrupt Enable. Read: Bridge Out Interrupt Enable Write: • '1' - enable interrupts • '0' - disable interrupts
1	Software Reset(1)	Read/Write	'0'	Software Reset. Read: This bit will always read '0' since it is reset whenever a '1' is written to it. Write: • '1' - reset the bridge • '0' - resume normal bridge operation
2 to C_DCR_DWIDTH-1	Unused			

Notes:
1. The software reset will reset the entire PLB2OPB Bridge regardless of the current PLB and OPB transaction states. Therefore, the software reset should be used with caution.

PLB to OPB Bridge Interrupt Description

The PLB to OPB Bridge has one interrupt request output called Bus_Error_Det. This interrupt is an edge type interrupt and is automatically reset to the inactive state on the next clock cycle, therefore an explicit interrupt acknowledge is not required.

The active level of the Bus_Error_Det interrupt is determined by the design parameter, C_IRQ_ACTIVE.

Note that if interrupts are enabled, then an interrupt request from the PLB to OPB Bridge will be generated whenever any bus error is detected regardless of whether masters have locked their error fields or not.

Also note that if the parameter C_DCR_INTFCE is 0 indicating that there is no DCR interface, then interrupts will be still be enabled as the default state of the Interrupt Enable bit in the BGO_CTRL_REG is asserted.

PLB to OPB Bridge Block Diagram

High-Level Description

Figure 9 provides a high-level overview of the design of the PLB to OPB Bridge. PLB transactions are received and decoded in the PLB interface logic.

The PLB interface logic then generates the necessary sequence of OPB signals (including address, byte enables, and data) to perform the transaction. These OPB signal patterns are then loaded into the PLB-to-OPB transfer interface logic which acts like a FIFO to de-couple the PLB interface from the OPB master.

The OPB interface logic receives the data patterns and outputs them on OPB. The OPB logic also handles the OPB hand-shaking signals such as request/grant, select, retry, time-out, and transfer acknowledge. For PLB read transactions, the OPB master returns the read data to the PLB logic via the OPB-to-PLB receive data interface.

The receive data interface also acts like a FIFO to de-couple the data return path and status flags from the OPB to the PLB logic.

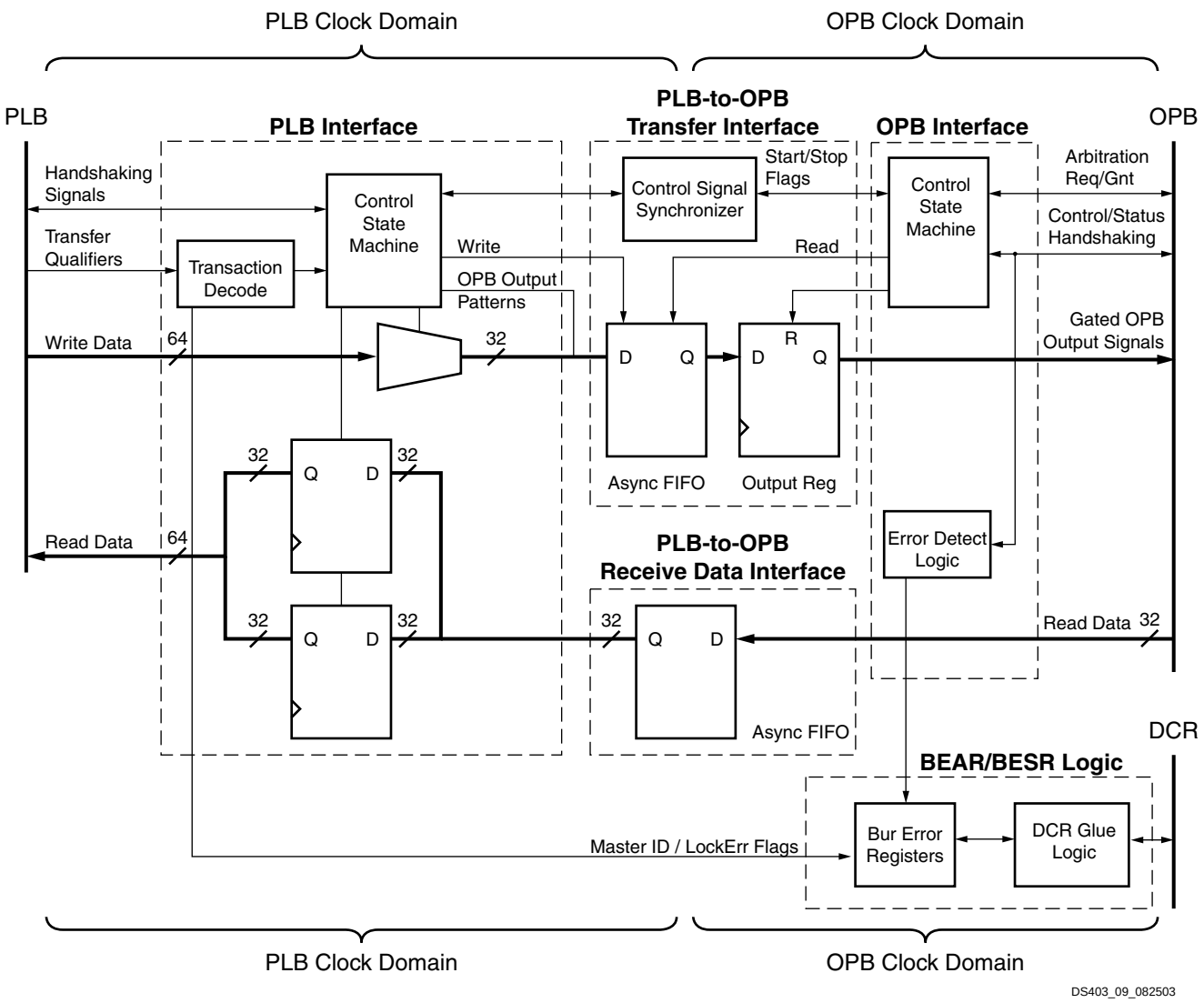

Figure 9: **High-Level Overview of the PLB to OPB Bridge**

PLB Interface

The PLB interface is designed with a pipelined architecture to improve timing and to support high clock frequencies. Output signals to the PLB are designed to be driven out from flip-flops through a minimal amount of logic.

PLB input signals are also limited to a few levels of logic before being registered. Pipelining introduces some additional latency in the design since some signals are delayed through registers.

However, the use of pipelining balances low transaction latency with high clock frequency timing.

Address Decode Cycle

PLB transactions begin with an address decode cycle. An extra clock cycle is used to decode the address and transfer qualifiers of a PLB request before the address can be acknowledged.

This address decode cycle can be performed during a secondary address request (**PLB_SAValid** = 1) to help overlap the address decode cycle with a previous transaction.

Though address decoding can be performed during secondary address cycles, address acknowledge (**BGO_addrAck**) is only returned during a primary address cycle (**PLB_PAValid** = 1).

You can set address decoding parameters to configure the upper and lower address boundary of up to four separate address ranges. If a particular address range needs to support 16 word cacheline transactions, the base address of this address range should be aligned to a 64-byte boundary.

If a particular address range contains guarded memory, the high address of this address range should be aligned to a 1K-byte boundary .Note that the base address and high address of each address range should be set so as few upper address bits as possible need to be compared. Reducing the size of the address comparators improves performance and reduces logic utilization.

Address Sequencing

The PLB transfer is then sequenced into sets of OPB transactions. Each set of OPB transaction is formed into a data pattern consisting of the OPB write data, OPB address, OPB byte enables, read/write flag, bus lock flag, sequential address flag, and end-of-transfer flag.

A state machine (based on a down counter) is loaded with the length of the data transfer and then decremented until it reaches zero. This controls the generation of OPB signals. For each non-zero counter value, an OPB address counter is incremented, while additional logic sets the necessary OPB transfer qualifiers.

The OPB transaction patterns are then loaded in a FIFO memory for processing by the OPB logic. The organization and bit definitions of these data patterns are shown in Table 13.

Table 13: **PLB-to-OPB Transfer Data Bit Definitions**

Bits	Description
0 to 31	OPB write data
32 to 63	OPB address
64 to 67	OPB byte enables
68	OPB read not write
69	OPB bus lock
70	OPB sequential address
71	Last word

Write Transactions

For write transactions, the write data acknowledges (**BGO_wrDAck**) are returned over PLB and the write data (**PLB_wrDBus(0:C_PLB_DWIDTH-1)**) is registered and sent on to the FIFO memory. The bridge can buffer up to 16 words of write data into the FIFO.

Both 32- and 64-bit PLB transactions are supported while OPB transactions are fixed at 32 bits wide. Therefore, the PLB interface logic is aware of the master size (**PLB_MSize[0:1]**) and multiplexes data down to a 32-bit data path as necessary.

After all the write data has been transferred over, the write complete (**BGO_wrComp**) signal can be asserted. Though the transaction is complete over PLB, the bridge will wait for the OPB logic to report that the transaction was completed over the OPB before it can begin processing another transaction.

If another PLB transaction is requested before the OPB transaction has completed, **BGO_rearbitrate** will be asserted to indicate that the bridge is unable to process the requested PLB transaction.

Read Transactions

For read transactions, the OPB write data signals are set to zero while the other OPB address and transfer qualifiers are generated (in the same manner as with write transactions). Read data acknowledges (**BGO_rdDAck**) and read data (**BGO_rdDBus(0:C_PLB_DWIDTH-1)**) are returned over the PLB as the read data is received from the OPB interface.

The PLB read complete signal (**BGO_rdComp**) is asserted when all the read data has been returned or when the master terminates a burst cycle. Read requests are generated for up to 16 words during read transactions.

For variable length bursts where the burst length is not specified, the bridge will pre-fetch up to 16 words of data to satisfy the read request. For fixed-length bursts, the bridge will fetch only the words requested, up to 16 words. Cacheline read data transfers are only sequenced in linear order instead of *target word first* order.

PLB-to-OPB Transfer Interface

This group of logic implements a FIFO interface between the PLB and OPB interfaces. It also synchronizes and passes handshaking flags between the two sides to communicate that the FIFO is not empty or to signal that a PLB read burst has been terminated by the PLB master.

This requires that the rising edges of the OPB clock and the PLB clock be phase aligned. LUT-based dual-port memories are used as the memory elements to store the data, since they are a dual-port memory that can support independent clocks on each side.

OPB Interface

The OPB interface is designed with a pipelined architecture to improve timing and support high clock frequencies. Output signals to OPB are designed to be driven out from flip-flops through a minimal amount of logic.

OPB input signals are also limited to a few levels of logic before being registered. To further improve timing, output signals to the OPB are internally gated off.

The OPB interface begins an OPB transaction when it receives a start flag indicating that the FIFO is not empty. It requests (**BGO_request**) access to the OPB bus and begins the transaction when a grant (**OPB_MnGrant**) is received.

The OPB transaction is performed by simply driving the contents of the FIFO out onto OPB and advancing the FIFO read pointer after every transfer acknowledge (**OPB_xferAck**). After all the data in the FIFO has been transmitted (indicated by the end-of-transfer flag being high), the OPB logic goes back to its IDLE state and waits for the FIFO to leave the empty state before beginning a new transaction.

During read transactions, data is returned to the PLB logic by way of the OPB to PLB receive data interface.

Note that the PLB to OPB Bridge will wait C_BGI_TRANSABORT_CNT clocks after asserting **BGO_request** to receive its grant. If **OPB_MnGrant** is not received in this time period, the PLB to OPB bridge signals the OPB to PLB bridge to abort its current operation. Please **Bridge to Bridge Communication - Deadlock Prevention** for details.

All data transfers greater than one word are performed as locked transfers over OPB (**BGO_busLock** = 1). To improve bus efficiency, the last word in a set of OPB transfers is unlocked as recommended by the OPB specification.

If the PLB bus is locked for a transaction and remains locked (**PLB_busLock** asserted) after the bridge completes all its OPB data transfers, the OPB master will hold **BGO_busLock** until the PLB is unlocked. This allows a series of *locked PLB transfers* to be carried out as a series of *locked OPB transfers*.

All bursts transfers greater than one word and all cacheline transfers are performed with the sequential address flag asserted (**BGO_seqAddr** = 1). The sequential address flag helps improve the performance of OPB slave peripherals.

All sequential bursts are also performed with bus lock asserted (as required by the OPB specification). Note that non-burst, 5 to 8 byte transfers from 64-bit PLB masters are performed as two OPB transfers, but the sequential address flag is not asserted.

If an OPB time-out occurs while the OPB master is driving the bus, it backs off the bus for one cycle as required by the OPB specification, but maintains **BGO_busLock**.

This allows it to advanced the FIFO read pointer and continue the transaction. A time-out condition causes an error flag to be passed over to the PLB logic (to drive **BGO_MErr(0:C_PLB_NUM_MASTERS-1)**) and to the BEAR/BESR logic to update the error registers.

Slaves that generate an **OPB_retry** cause the OPB master to back off the bus for one cycle before attempting the transaction over again by requesting access to the bus. This will be continued until successful completion of the transaction.

OPB error acknowledge signals (**OPB_errAck**) generate error flags similar to time-out error flags that are sent to the PLB and BEAR/BESR logic. Error acknowledge signals are otherwise treated like transfer acknowledge signals in that the OPB master continues the data transfer with the next data.

Reception of either **OPB_errAck** or **OPB_timeout** will cause the assertion of the **Bus_Error_Det** interrupt provided the design has been parameterized to include the DCR interface (**C_DCR_INTFCE**=1) and interrupts have been enabled.

OPB-to-PLB Receive Data Interface

This module acts like a FIFO for data going back from OPB to PLB. It communicates read data and status flags back to OPB and synchronizes signals passing between the two clock domains. This requires that the rising edges of the OPB clock and the PLB clock be phase aligned. The organization and bit definitions of the OPB-to-PLB data are shown in Table 14.

Table 14: **OPB-to-PLB Transfer Data Bit Definitions**

Bits	Description
0 to 31	OPB read data
32 to 35	OPB address bits 26 to 29
36	OPB errAck or OPB timeout
37	Last word

BEAR/BESR Logic (C_DCR_INTFCE=1)

When the design has been parameterized to include the DCR interface (**C_DCR_INTFCE**=1), error flags from the OPB interface as well as the **PLB_masterID(0:C_PLB_MID_WIDTH-1)** and **PLB_lockErr** qualifiers are forwarded to the BEAR/BESR logic to record errors. The error registers are accessible over the DCR bus.

They record information such as the type of error (error acknowledge versus time-out), whether it was a read or write, and which PLB master caused the error. The address and byte enable patterns corresponding to the error are also recorded. If the **PLB_lockErr** signal was asserted, the corresponding PLB master's BEAR/BESR is locked so that subsequent errors do not overwrite it, unless the error register is first cleared. If **PLB_lockErr** is not asserted, then the error information can be overwritten by a subsequent error.

The DCR bus provides access to the BEAR/BESR and operates on the same clock as the OPB logic. This requires that the DCR master's clock be synchronous to the OPB clock in order for the DCR slave interface to work properly.

Whenever an error has been detected, the **Bus_Error_Det** interrupt is asserted. See PLB to OPB Bridge Interrupt Description for details on this interrupt.

Bridge to Bridge Communication - Deadlock Prevention

When a PLB master is requesting a read on the OPB and an OPB master is requesting a read on the PLB, there is a possibility that the PLB to OPB bridge can't access the OPB and the OPB to PLB bridge can't access the PLB, thus resulting in deadlock.

This is due to the fact that the OPB to PLB bridge suppresses the OPB time-out while it waits for data from the PLB. However, the PLB is busy waiting for data from the OPB. Since the PLB doesn't have a time-out mechanism during the data phase of a transaction, both busses are stalled and deadlock can occur.

To prevent this situation, the PLB to OPB bridge will wait C_BGI_TRANSABORT_CNT OPB clock cycles after asserting its OPB request for its grant to be asserted. If the grant is not asserted in this time period, the PLB to OPB bridge will assert **BGI_Trans_Abort** to the OPB to PLB bridge.

This indicates that the OPB to PLB bridge should issue a retry to the OPB master on the OPB which will cause an OPB arbitration cycle, allowing the PLB to OPB bridge a chance to gain control of the OPB. Assertion of **BGI_Trans_Abort** will also cause the OPB to PLB bridge to abort its current PLB transaction.

Note that this signal will not help the PLB to OPB Bridge gain control of the OPB when the OPB is busy with transactions between other peripherals on the OPB.

Bridge to Xilinx OPB IPIF Communication - Deadlock Prevention

When a PLB master is requesting a read of an OPB slave that utilizes the Xilinx Master/Slave OPB IPIF, there is a possibility that infinite retries can occur if the OPB IPIF Master is requesting a read of a PLB slave. This can result in a deadlock condition for the PLB master making the read request of the OPB slave, the OPB Master/Slave device, the OPB2PLB bridge making read requests on the PLB, and all PLB read operations.

This is true for PLB2OPB bridges that are designed to be compatible with the IBM separate master and slave OPB device architecture when used with the Xilinx OPB Master/Slave IPIF architecture. Xilinx PLB2OPB bridges before version 1.01.a where designed for the IBM separate master and slave OPB device architecture.

The problem arises due to the fact that the Xilinx OPB IPIF Master/Slave architecture uses both master and slave modules for all master transactions. If the local master is attempting to read from a slave, then the OPB IPIF will retry all requests made to the local slave module because, in certain conditions, the local master has taken control of the slave module. However, if the PLB2OPB bridge is designed for the IBM separate Master and Slave device architecture the PLB read bus is dedicated to the PLB2OPB bridge completing the read operation of the OPB slave. This because an IBM architecture-based PLB2OPB bridge will issue an AddrAck immediately which essentially locks the PLB read bus until data is transferred.

But if the read is targeting an OPB device using the Xilinx OPB Master/Slave IPIF and the OPB device local master is attempting to read data from a PLB slave, then the PLB2OPB bridge read request will be retried forever on the OPB. Furthermore, the local OPB master read request to the OPB2PLB bridge will also be retried forever due to the priority of the PLB2OPB bridge that is described in the previous section.

In this version of the PLB2OPB bridge, the PLB read transaction was redesigned to not assert AddrAck until either data is received from the OPB slave, an OPB retry is asserted, or an OPB timeout is asserted. With this design, the PLB2OPB bridge will get off the PLB read bus if an OPB retry is asserted. This opens a window of opportunity for the OPB2PLB bridge to complete a read transaction with a PLB slave. However, the length of time the window is open is determined by the PLB arbiter.

Xilinx PLB arbiters prior to the arbiter found in PLB V34 version 1.02.a did not have a sufficiently long window for the OPB2PLB bridge to get granted the PLB bus; consequently, this bridge must be used with PLB V34 version 1.02.a (and possibly later versions) to insure the deadlock can be broken. This bridge and the PLB arbiter pair provide compatibility with Xilnx's OPB Master/Slave architecture.

The new pair of PLB2OPB bridge and PLB arbiter have a new signal interconnecting them to provide the arbiter with information that a rearbitrate on PLB read has been asserted by a PLB2OPB bridge. This rearbitration signal qualifier signals the arbiter to open the window of opportunity for the OPB2PLB bridge to gain access to the PLB read bus. This window of opportunity is defined by a parameterized number of PLB clock periods. The parameter is an input to the PLB V34 core. See the PLB V34 documentation for guidelines on the correct parameter setting for your system.

Design Implementation

Device Utilization and Performance Benchmarks

Since the PLB to OPB 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 PLB to OPB Bridge is combined with other pieces of the FPGA design, the utilization of FPGA resources and timing of the PLB to OPB Bridge design will vary from the results reported here.

In order to analyze the PLB to OPB Bridge's timing within the FPGA, a design was created that instantiated the PLB to OPB Bridge with registers on all of the PLB to OPB Bridge inputs and outputs. This allowed a constraint to be placed on the clock net for the PLB to OPB Bridge to yield more realistic timing results.

The f_{MAX} parameter shown in Table 15 was calculated with registers on the PLB to OPB Bridge inputs and outputs. Note however, that the resource utilizations reported in Table 15 do not include the registers on the PLB to OPB Bridge inputs and outputs.

The PLB to OPB Bridge benchmarks shown in Table 15 are for a Virtex-II Pro -7 FPGA.

Table 15: **PLB to OPB Bridge FPGA Performance and Resource Utilization Benchmarks (Virtex-II Pro -7)**

Parameter Values					Device Resources			OPB f_{MAX} (MHz)	PLB f_{MAX} (MHz)
C_NUM_ADDR_RNG	C_RNG0_BASEADDR- C_RNG0_HIGHADDR C_RNG1_BASEADDR- C_RNG1_HIGHADDR C_RNG2_BASEADDR- C_RNG2_HIGHADDR C_RNG3_BASEADDR- C_RNG3_HIGHADDR	C_PLB_NUM_MASTERS	C_NO_PLB_BURST	C_DCR_INTFCE	Slices	Slice Flip-Flops	4-input LUTs	fMAX	fMAX
1	0x00000000-0x000000FF N/A N/A N/A	8	1	0	605	559	559	202	202
1	0x00000000-0x0000FFFF N/A N/A N/A	8	1	0	605	559	557	200	204
1	0x00000000-0x00FFFFFF N/A N/A N/A	8	1	0	604	559	555	197	203
1	0x00000000-0xFFFFFFFF N/A N/A N/A	8	1	0	602	559	552	205	201
2	0x00000000-0x3FFFFFFF 0x40000000-0x7FFFFFFF N/A N/A	8	1	0	602	559	552	189	202
3	0x00000000-0x3FFFFFFF 0x40000000-0x7FFFFFFF 0x80000000-0xCFFFFFFF N/A	8	1	0	602	559	552	203	204
4	0x00000000-0x3FFFFFFF 0x40000000-0x7FFFFFFF 0x80000000-0xBFFFFFFF 0xC0000000-0xFFFFFFFF	8	1	0	602	559	552	195	205

Table 15: **PLB to OPB Bridge FPGA Performance and Resource Utilization Benchmarks (Virtex-II Pro -7)**

1	0x00000000-0x00FFFFFF N/A N/A N/A	2	1	0	595	547	535	197	200
1	0x00000000-0x00FFFFFF N/A N/A N/A	4	1	0	599	551	543	197	200
1	0x00000000-0x00FFFFFF N/A N/A N/A	6	1	0	603	555	549	197	205
1	0x00000000-0x00FFFFFF N/A N/A N/A	16	1	0	620	575	580	205	205
1	0x00000000-0x00FFFFFF N/A N/A N/A	8	0	0	657	581	643	199	188
1	0x00000000-0x00FFFFFF N/A N/A N/A	8	0	1	836	812	823	159	187
1	0x00000000-0x00FFFFFF N/A N/A N/A	8	1	1	787	790	744	172	173
1	0x00000000-0x00FFFFFF N/A N/A N/A	8	1	0	688	578	680	199	204

Notes:

1. These benchmark designs contain only the PLB to OPB Bridge with registered inputs/outputs without any additional logic. Benchmark numbers approach the performance ceiling rather than representing performance under typical user conditions.
2. Device resource numbers do not include the registers for the PLB to OPB Bridge I/O.
3. Max frequency calculated with registers on the PLB to OPB Bridge I/O.

Reference Documents

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

- *IBM CoreConnect™ 64-Bit PLB to OPB Bridge Core User's Manual*
- *IBM CoreConnect 64-Bit Processor Local Bus: Architecture Specification*
- *IBM CoreConnect 64-Bit On-Chip Peripheral Bus: Architecture Specifications*
- *IBM CoreConnect 32-Bit Device Control Register Bus: Architecture Specifications*
- *Virtex-II Pro Platform FPGAs (Advance Product Specification)*

Revision History

Date	Version	Revision
04/22/02	1.0	Initial Xilinx release.
05/31/02	1.1	Update for EDK 1.0
07/09/02	1.2	Update for core version b
07/24/02	1.3	Add XCO parameters for System Generator
10/29/02	1.4	Updated resource utilization table for latest software revision
11/11/02	1.5	Updated parameter table to include reserved parameters (parameters not modifiable by user)
01/07/03	1.6	Update for EDK SP3
07/07/03	1.7	Update to new template
09/18/03	1.7.1	Update graphics to GSC standard and fix trademarks
04/16/04	1.7.2	Revison to v1_01_a. Redesigned PLB read operation to make system compatible with Xilinx Master/Slave OPB IPIF architecture.
06/07/04	1.8	Hardware revision to v1.01b
8/10/04	1.9	Updated trademarks and supported families listing; made minor formatting and content edits, and inserted cross-references for tables and figures.
7/15/05	2.0	Updated to incorporate CR207565.