Processor Local Bus (PLB) v4.6 (v1.04a)

DS531 April 24, 2009 **Product Specification**

Introduction

The Xilinx 128-bit Processor Local Bus (PLB) v4.6 provides bus infrastructure for connecting an optional number of PLB masters and slaves into an overall PLB system. It consists of a bus control unit, a watchdog timer, and separate address, write, and read data path units, as well as an optional DCR slave interface to provide access to its bus error status registers.

Features

- Arbitration support for a configurable number of PLB master devices
- PLB address and data steering support for all masters
- 128-bit, 64-bit, and 32-bit support for masters and slaves
- PLB address pipelining (supported in shared bus mode or point-to-point configuration)
- Three-cycle arbitration
- Four levels of dynamic master request priority
- Selectable round robin or fixed priority arbitration
- Configurable optimization for point-to-point topology
- PLB watchdog timer
- PLB architecture compatible
- Complete PLB bus structure provided
  - ♦ Supports a configurable number of slave devices
  - ♦ No external OR gates required for PLB slave input signals
- PLB Reset circuit
  - ♦ PLB Reset generated synchronously to the PLB clock
  - ♦ PLB Reset generated synchronously from external reset when external reset provided
  - ♦ Provides vectorized reset signal to reduce system fanout for improved timing
  - ♦ Active state of external reset selectable via a design parameter

LogiCORE™ Facts		
Core Specifics		
Supported Device Family	See EDK Supported Device Families.	
Version of Core	plb_v46	v1.04a
Resources Used	See Table 13, page 34.	
Provided with Core		
Documentation	Product Specification	
Design File Formats	VHDL	
Constraints File	N/A	
Verification	N/A	
Instantiation Template	N/A	
Design Tool Requirements		
Xilinx Implementation Tools	See Tools for requirements.	
Verification		
Simulation		
Synthesis		
Support		
Provided by Xilinx, Inc.		

Functional Description

The Xilinx PLB consists of a central bus arbiter, the necessary bus control and gating logic, and all necessary bus OR/MUX structures. The Xilinx PLB provides the entire PLB bus structure and allows for direct connection with a configuration number of masters and slaves. Figure 1 provides an example of the PLB connections for a system with three masters and three slaves.

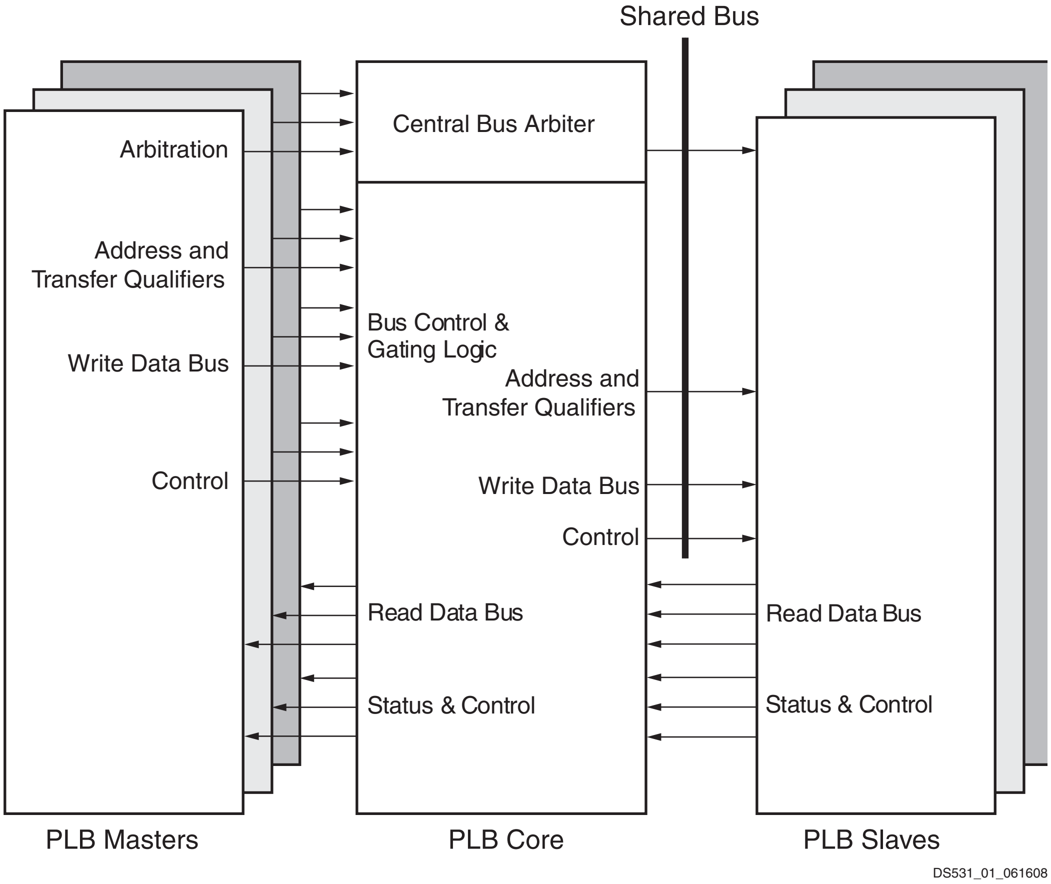


Figure 1: **PLB Interconnect Diagram**

Basic Operation

The Xilinx PLB has 3-cycle arbitration during the address phase of the transaction as shown in Figure 2. There is a two-cycle delay from Mn_request to PLB_PAValid. If the slave can respond combinatorially in the same cycle—the optimistic assumption shown and theoretically possible for a write transaction—the whole transaction takes three cycles.

The two-cycle delay from Mn_request to PLB_PAValid holds for the case where there are two or more attached masters and allows one cycle for a priority arbitration to occur and one cycle to route the selected master's transaction data and qualifiers to the slaves. If there is a single master, arbitration is not necessary and transaction data and qualifiers can be driven to the slaves without multiplexing. This allows PLB_PAValid to be driven after one clock, saving a cycle of latency.

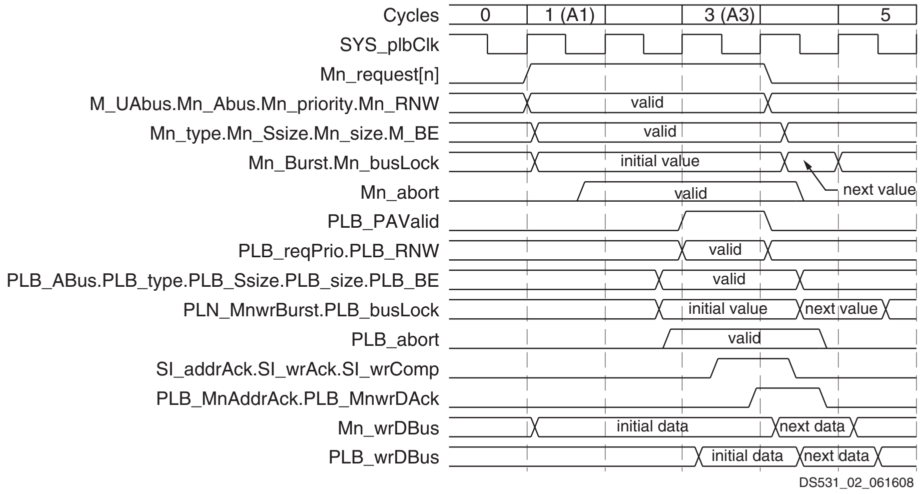


Figure 2: **3-cycle arbitration (Xilinx Implementation)**

During the bus arbitration cycle (in a fixed priority arbitration scheme), the bus arbitration control unit uses two priority bits from each requesting master to determine which master is granted the bus. The two bits **M_priority[i*2:i*2+1]** are the priority bits for master **i**. The two priority bits are interpreted as an integer between 0 and 3—with the bit at the lowest index having the highest weight of two. The value of zero represents the lowest priority. From there, priority increases with increasing numerical value.

In addition, the Xilinx PLB arbitration logic supports the fixed priority scheme to handle "tie" situations (that is, situations when two or more masters request the bus simultaneously while presenting the same level of request priority). Selection of the priority mode during tie situations is shown in Table 1 where n = C_PLBV46_NUM_MASTERS.

Table 1: **Priority Order for Bus Masters**

Highest Priority	Decreasing Priority			Lowest Priority
Master $_0$	Master $_1$	...	Master $_{n-2}$	Master $_{n-1}$

In the round robin arbitration scheme, the bus arbitration control unit allows the least recently used master to win arbitration and control the bus. Once a master is granted the bus, it will then become the lowest priority master in the next arbitration cycle. Configuring the PLB in a round robin scheme allows each master in the system an opportunity to be granted the bus. This is critical in system configurations where certain masters can lock out other masters from being granted the bus simply due to connection ordering on the PLB. Round robin arbitration prevents the need of each master to increase the M_priority bits to the highest level in order for a chance to win arbitration on the bus.

Address Pipelining

The read and write data buses of the PLB can be concurrently active. Therefore, two *primary transactions*, one read and one write, are launched sequentially and completed in parallel. Beyond this, the PLB protocol allows for additional *secondary transactions* to be acknowledged through the address phase and queued up, waiting to complete the data transfer when signaled to that the needed data bus is free. The PLB v4.6 protocol allows for such *address pipelining* to be arbitrarily deep, but does not require it.

To make use of address pipelining, the Xilinx PLB must incorporate the implied extra state and bookkeeping logic *and* slaves must be designed to accept secondary transactions. If support is missing in either place, all transactions proceed as primary transactions and the expense associated with implementing the unused secondary-transaction capability--whether in the Xilinx PLB or slaves--is wasted.

The Xilinx PLB is introduced into an FPGA embedded computing environment where slaves generally do not support secondary addresses; however, the MicroBlaze™ processor does support this feature. Therefore, the Xilinx PLB will support address pipelining by default. However, there is an option to disable address pipelining, when secondary address support is not needed. The option is disabled by setting C_ADDR_PIPELINING_TYPE=0, which disables two-deep address pipelining (one primary and one secondary transaction for each of read and write). Deeper address pipelining is not supported. To get the default of supporting address pipelining, set C_ADDR_PIPELINING_TYPE=1.

PLB two-deep address pipelining is not limited to the shared bus mode configuration. When the PLB is configured in a point-to-point topology, the two-deep address pipeline also is enabled. To enable this, both the parameters must be set as C_P2P=1 and C_ADDR_PIPELINING_TYPE=1. The point-to-point configuration with address pipelining allows slave devices to assert the Sl_addrAck to the asserted PLB_SAValid, secondary address valid. On read transactions, the slave must monitor the PLB_rdPrim signal which indicates the primary transaction data phase is complete and the slave can begin to drive Sl_rdDAck and Sl_rdComp for the secondary transaction which was previously acknowledged. Figure 3 illustrates this usage case.

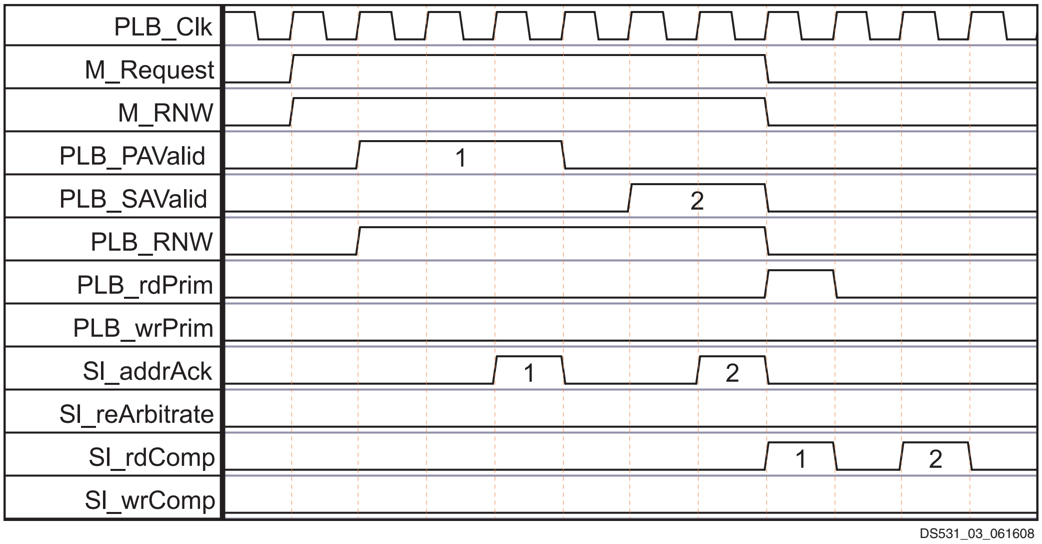


Figure 3: **Point-to-point Address Pipelining (AddrAck to SAValid)**

Figure 4 illustrates an example of a delayed Sl_addrAck assertion to the secondary address valid indicator. In this case, the PLB_SAValid will be promoted to the PLB_PAValid if the primary transaction completes (with the assertion of Sl_rdComp) prior to the Sl_addrAck.

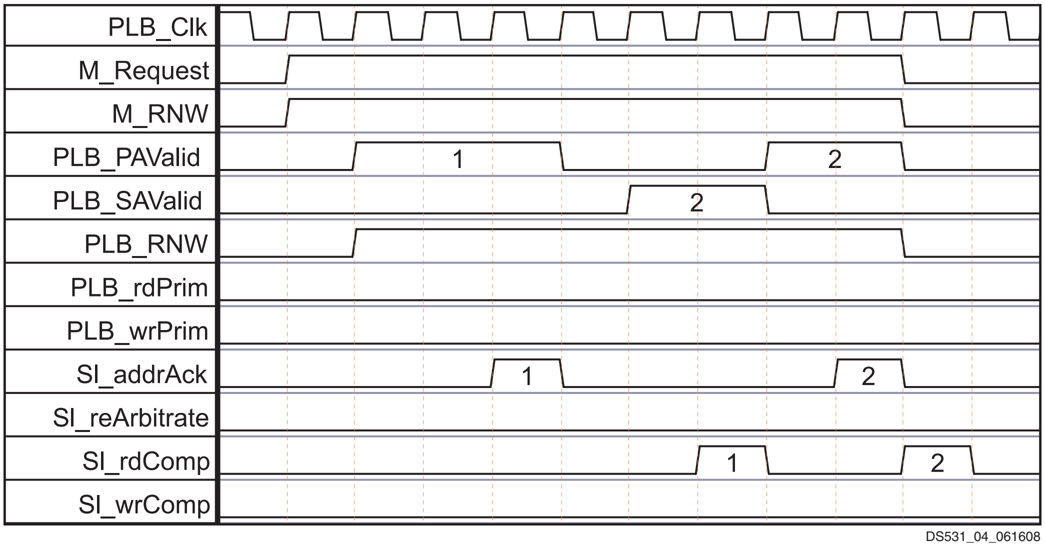


Figure 4: **Point-to-point Address Pipelining (Secondary to Primary Promotion)**

Figure 5 illustrates back to back read requests followed by a subsequent write request of the master. With separate read and write data buses on the PLB, the arbiter is capable of asserting PAValid for a subsequent write operation prior to the previous read operations completing.

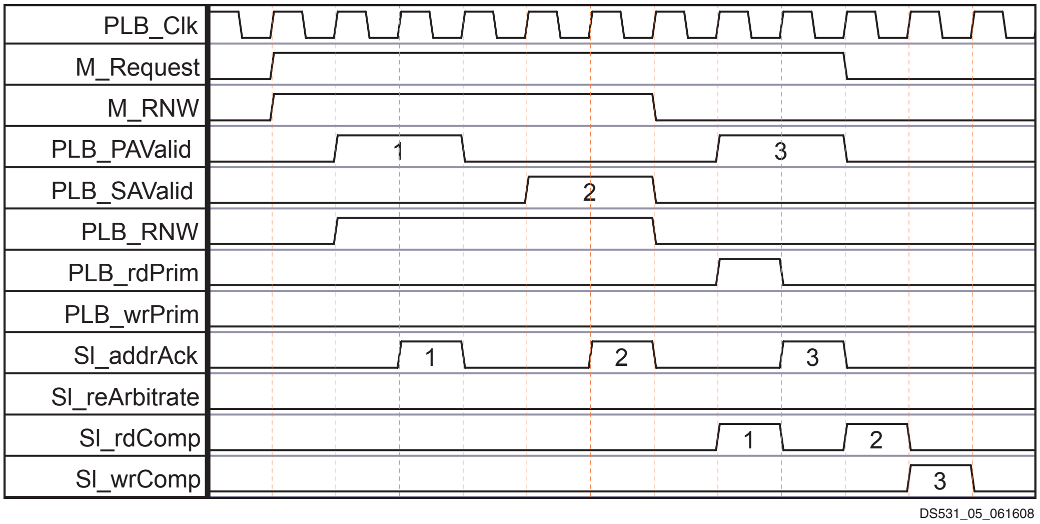
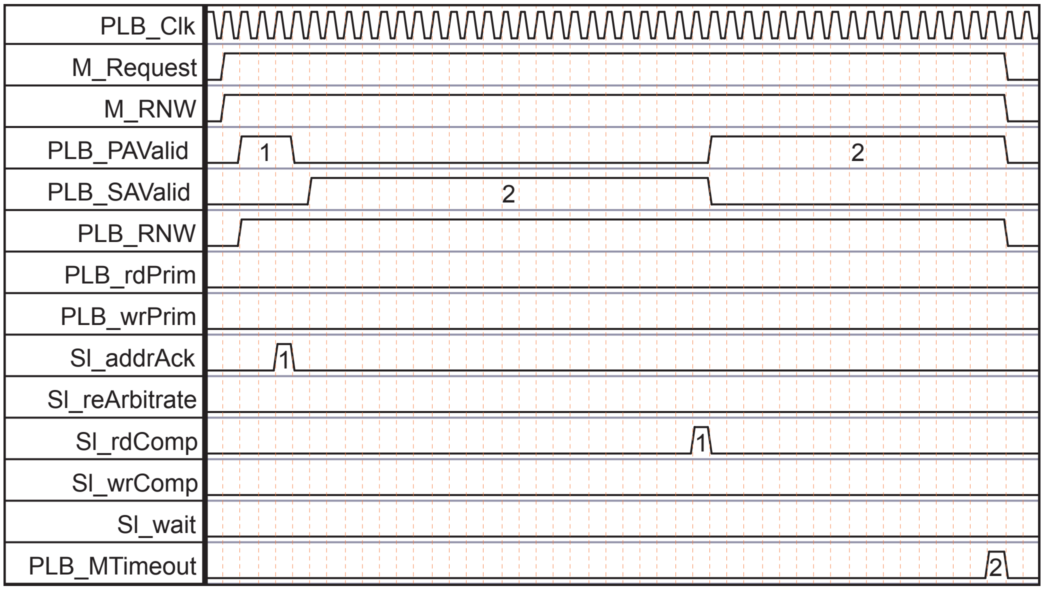


Figure 5: **Pipelined Read Transactions with Subsequent Write**

Slave devices supporting address pipelining must be able to assert either Sl_reArbitrate or Sl_addrAck to the PLB_SAValid transaction. The PLB_SAValid operation cannot time out by the arbiter, only the primary transaction, indicated by the assertion of PLB_PAValid. Figure 6 illustrates an example of PLB_SAValid asserted for more than the primary transaction time out count value. The arbiter will not time out the PLB_SAValid transaction, but will wait for the promotion of PLB_SAValid to PLB_PAValid (indicated by Sl_rdComp), then the time out counter will be activated. If during the assertion of PLB_PAValid neither Sl_reArbitrate or Sl_addrAck is asserted, the PLB_MTimeout will be asserted after 16 clock cycles.



DS531_06_061608

Figure 6: **Timeout Only on Promotion of Secondary to Primary Transaction**

PLB Design Parameters

To allow for a PLB that is tailored to the target embedded system, certain features can be parameterized in the Xilinx PLB. This allows the user to have a design that only utilizes the resources required by the system and runs at the best possible performance. The features that can be parameterized in the Xilinx PLB are shown in Table 2.

Table 2: PLB Design Parameters

Generic	Feature / Description	Parameter Name	Allowable Values	Default Value	VHDL Type
PLB Features					
G1	Number of PLB Masters	C_PLBV46_NUM_ MASTERS	1 - 16(1)	4	integer
G2	Number of PLB Slaves	C_PLBV46_NUM_ SLAVES	1 - 16(2)	8	integer
G3	PLB Address Bus Width	C_PLBV46_AWIDTH	32	32	integer
G4	PLB Data Bus Width	C_PLBV46_DWIDTH	32, 64, 128	64	integer

Table 2: PLB Design Parameters *(Cont'd)*

Generic	Feature / Description	Parameter Name	Allowable Values	Default Value	VHDL Type
G5	Include DCR interface	C_DCR_INTFCE	1 = Include DCR slave interface; shared bus configuration only 0 = DCR slave interface not included; only allowed value in P2P configuration	0	integer
DCR Interface (Available only in a shared bus configuration)					
G6	DCR Base Address	C_BASEADDR	Valid DCR address[3]	None[4]	std_logic_vector
G7	DCR High Address	C_HIGHADDR	Valid DCR address	None	std_logic_vector
G8	DCR Address Bus Width	C_DCR_AWIDTH	10	10	integer
G9	DCR Data Bus Width	C_DCR_DWIDTH	32	32	integer
Interrupts					
G10	Active Interrupt State[5]	C_IRQ_ACTIVE	0 = interrupt request is driven as a falling edge 1 = interrupt request is driven as a rising edge	1	std_logic
System					
G11	Active level of external reset	C_EXT_RESET_HIGH	1 = external reset is active high 0=external reset is active low	1	integer
Auto-calculated parameters[6]					
G12	Number of bits required to encode the number of PLB Masters	C_PLBV46_MID_WIDTH	1 - log2(C_PLBV46_NUM_MASTERS)	2	integer
System					
G13	Target FPGA family	C_FAMILY	See C_FAMILY parameter values.		string
G14	Address pipelining type supported	C_ADDR_PIPELINING_TYPE	0 = no address pipelining 1 = 2-level address pipelining	1	integer

Table 2: PLB Design Parameters *(Cont'd)*

Generic	Feature / Description	Parameter Name	Allowable Values	Default Value	VHDL Type
G15	Optimize PLB for point-to-point topology (one master & one slave)	C_P2P	0 = PLB is configured in a shared bus mode topology 1 = PLB is configured with one master and one slave for point-to-point topology	0	integer
G16	Selects the arbitration scheme for all master devices connected to the bus.	C_ARB_TYPE	0 = Fixed priority 1 = Round robin	0	integer

Notes:

1. The supported allowable values for the parameter, C_PLBV46_NUM_MASTERS, are 1 - 16. Only the values of 1 - 8 have been tested in a unit-level verification environment.
2. The supported allowable values for the parameter, C_PLBV46_NUM_SLAVES, are 1 - 16. Only the values of 1 - 8 have been tested in a unit-level verification environment.
3. The range specified by C_BASEADDR and C_HIGHADDR must comprise a complete, contiguous power of two range such that range = 2^n, and the n least significant bits of C_BASEADDR must be zero. To allow for the 8 DCR registers within the PLB, n must be at least 3. Note that the DCR interface is only available in shared bus configuration; it is not available in P2P configuration.
4. No default value is specified for C_BASEADDR or C_HIGHADDR to insure that the actual value is set. For example, if the value is not set, a compiler error is generated.
5. The interrupt request output is generated as an edge type interrupt. A specific interrupt acknowledge response is not required.
6. These parameters are automatically calculated by the system generation tool and are not input by the user.

Allowable Parameter Combinations

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

To allow for the registers in the PLB design, this range must be at least 7; therefore n must be at least 3. This means that at a minimum, the three least significant bits of C_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 the user to trade-off address space resolution with size and speed of the PLB.

Some parameters can cause other parameters to be irrelevant. See Table 4 for information on the relationship between design parameters.

PLB I/O Signals

Table 3 provides a summary of all Xilinx PLB input/output (I/O) signals, the interfaces under which they are grouped, and a brief description of the signal.

Table 3: **PLB Pin Descriptions**

Port	Signal Name	Interface	I/O	Init State	Description
DCR Signals (Only available in shared bus configuration)					
P1	DCR_ABus[0:C_DCR_AWIDTH-1]	DCR	I		CPU DCR address bus
P2	DCR_Read	DCR	I		CPU read from DCR indicator
P3	DCR_Write	DCR	I		CPU write to DCR indicator
P4	DCR_DBus[0:C_DCR_DWIDTH-1]	DCR	I		DCR write data bus
P5	PLB_dcrAck	DCR	O	0	PLB DCR data transfer acknowledge
P6	PLB_dcrDBus[0:C_DCR_DWIDTH-1]	DCR	O	0	PLB DCR read data bus
PLB Status Signals					
P7	PLB_rdPendPri[0:1]	Master/Slave	O	0	PLB pending read request priority
P8	PLB_wrPendPri[0:1]	Master/Slave	O	0	PLB pending write request priority
P9	PLB_rdPendReq	Master/Slave	O	0	PLB pending bus read request indicator
P10	PLB_wrPendReq	Master/Slave	O	0	PLB pending bus write request indicator
P11	PLB_reqPri[0:1]	Master/Slave	O	0	PLB current request priority
Master Signals					
P12	M_abort[0:C_PLBV46_NUM_MASTERS-1]	Master	I		Master abort bus request indicator
P13	M_ABus[0:C_PLBV46_NUM_MASTERS*32-1]	Master	I		Master address bus, lower 32 bits for each master
P14	M_BE[0:C_PLBV46_NUM_MASTERS*C_PLBV46_DWIDTH/8-1]	Master	I		Master byte enables
P15	M_busLock[0:C_PLBV46_NUM_MASTERS-1]	Master	I		Master bus lock
P16	M_TAttribute[0:16*C_PLBV46_NUM_MASTERS-1]	Master	I		Master transfer attributes
P17	M_lockErr[0:C_PLBV46_NUM_MASTERS-1]	Master	I		Master lock error indicator
P18	M_mSize[0:C_PLBV46_NUM_MASTERS*2 -1]	Master	I		Master data bus port width
P19	M_priority[0:C_PLBV46_NUM_MASTERS*2 -1]	Master	I		Master bus request priority

Table 3: **PLB Pin Descriptions *(Cont'd)***

Port	Signal Name	Interface	I/O	Init State	Description
P20	M_rdBurst[0:C_PLBV46_NUM_ MASTERS-1]	Master	I		Master burst read transfer indicator
P21	M_request[0:C_PLBV46_NUM_ MASTERS-1]	Master	I		Master bus request
P22	M_RNW[0:C_PLBV46_NUM_ MASTERS-1]	Master	I		Master read not write
P23	M_size[0:C_PLBV46_NUM_ MASTERS*4 -1]	Master	I		Master transfer size
P24	M_type[0:C_PLBV46_NUM_ MASTERS*3-1]	Master	I		Master transfer type
P25	M_wrBurst[0:C_PLBV46_ NUM_MASTERS-1]	Master	I		Master burst write transfer indicator
P26	M_wrDBus[0:C_PLBV46_NUM_ MASTERS*C_PLBV46_ DWIDTH -1]	Master	I		Master write data bus
P27	PLB_MAddrAck[0:C_PLBV46_ NUM_MASTERS-1]	Master	O	0	PLB Master address acknowledge
P28	PLB_MBusy[0:C_PLBV46_NUM _MASTERS-1]	Master	O	0	PLB Master slave busy indicator
P29	PLB_MRdErr[0:C_PLBV46_ NUM_MASTERS-1]	Master	O	0	PLB Master slave read error indicator
P30	PLB_MWrErr[0:C_PLBV46_ NUM_MASTERS-1]	Master	O	0	PLB Master slave write error indicator
P31	PLB_MRdBTerm[0:C_PLBV46_N UM_MASTERS-1]	Master	O	0	PLB Master terminate read burst indicator
P32	PLB_MRdDAck[0:C_PLBV46_ NUM_MASTERS-1]	Master	O	0	PLB Master read data acknowledge
P33	PLB_MRdDBus[0:C_PLBV46_N UM_MASTERS*C_PLBV46_ DWIDTH -1]	Master	O	0	PLB Master read data bus
P34	PLB_MRdWdAddr[0:C_PLBV46_ NUM_MASTERS*4 -1]	Master	O	0	PLB Master read word address
P35	PLB_MRearbitrate[0:C_PLBV46_ NUM_MASTERS-1]	Master	O	0	PLB Master bus rearbitrate indicator
P36	PLB_MSSize[0:C_PLBV46_ NUM_MASTERS*2 -1]	Master	O	0	PLB Master slave data bus port width
P37	PLB_MWrBTerm[0:C_PLBV46_N UM_MASTERS-1]	Master	O	0	PLB Master terminate write burst indicator
P38	PLB_MWrDAck[0:C_PLBV46_ NUM_MASTERS-1]	Master	O	0	PLB Master write data acknowledge
P39	PLB_MTimeout[0:C_PLBV46_ NUM_MASTERS-1]	Master	O	0	PLB address-phase timeout indicator
Slave Signals					
P40	PLB_abort	Slave	O	0	PLB abort bus request indicator

Table 3: **PLB Pin Descriptions *(Cont'd)***

Port	Signal Name	Interface	I/O	Init State	Description
P41	PLB_ABus[0:31]	Slave	O	0	PLB address bus, lower 32 bits
P42	PLB_BE[0:C_PLBV46_DWIDTH/ 8-1]	Slave	O	0	PLB byte enables
P43	PLB_busLock	Slave	O	0	PLB bus lock
P44	PLB_TAttribute[0:15]	Slave	O	0	PLB transfer attribute
P45	PLB_lockErr	Slave	O	0	PLB lock error indicator
P46	PLB_masterID[0:C_PLBV46_MID_ WIDTH-1]	Slave	O	0	PLB current master identifier
P47	PLB_MSize[0:1]	Slave	O	0	PLB data bus port width indicator
P48	PLB_PAValid	Slave	O	0	PLB primary address valid indicator
P49	PLB_rdBurst	Slave	O	0	PLB burst read transfer indicator
P50	PLB_rdPrim[0:C_PLBV46_NUM_SLAVES-1]	Slave	O	0	PLB secondary to primary read request indicator
P51	PLB_RNW	Slave	O	0	PLB read not write
P52	PLB_SAValid	Slave	O	0	PLB secondary address valid
P53	PLB_size[0:3]	Slave	O	0	PLB transfer size
P54	PLB_type[0:2]	Slave	O	0	PLB transfer type
P55	PLB_wrBurst	Slave	O	0	PLB burst write transfer indicator
P56	PLB_wrDBus[0:C_PLBV46_DWIDTH-1]	Slave	O	0	PLB write data bus
P57	PLB_wrPrim[0:C_PLBV46_NUM_ SLAVES-1]	Slave	O	0	PLB secondary to primary write request indicator
P58	Sl_addrAck[0:C_PLBV46_NUM_SLAVES-1]	Slave	I		Slave address acknowledge
P59	Sl_MRdErr[0:C_PLBV46_NUM_SLAVES*C_PLBV46_NUM_MASTERS-1]	Slave	I		Slave read error indicator
P60	Sl_MWrErr[0:C_PLBV46_NUM_SLAVES*C_PLBV46_NUM_MASTERS-1]	Slave	I		Slave write error indicator
P61	Sl_MBusy[0:C_PLBV46_NUM_SLAVES*C_PLBV46_NUM_MASTERS-1]	Slave	I		Slave busy indicator
P62	Sl_rdBTerm[0:C_PLBV46_NUM_SLAVES-1]	Slave	I		Slave terminate read burst transfer
P63	Sl_rdComp[0:C_PLBV46_NUM_SLAVES-1]	Slave	I		Slave read transfer complete indicator
P64	Sl_rdDAck[0:C_PLBV46_NUM_SLAVES-1]	Slave	I		Slave read data acknowledge
P65	Sl_rdDBus[0:C_PLBV46_NUM_SLAVES*C_PLBV46_DWIDTH-1]	Slave	I		Slave read data bus

Table 3: **PLB Pin Descriptions *(Cont'd)***

Port	Signal Name	Interface	I/O	Init State	Description
P66	Sl_rdWdAddr[0:C_PLBV46_NUM_ SLAVES*4-1]	Slave	I		Slave read word address
P67	Sl_rearbitrate[0:C_PLBV46_NUM_ SLAVES-1]	Slave	I		Slave rearbitrate bus indicator
P68	Sl_SSize[0:C_PLBV46_NUM_ SLAVES*2-1]	Slave	I		Slave data bus port size indicator
P69	Sl_wait[0:C_PLBV46_NUM_ SLAVES-1]	Slave	I		Slave wait indicator
P70	Sl_wrBTerm[0:C_PLBV46_NUM_ SLAVES-1]	Slave	I		Slave terminate write burst transfer
P71	Sl_wrComp[0:C_PLBV46_NUM_ SLAVES-1]	Slave	I		Slave write transfer complete indicator
P72	Sl_wrDAck[0:C_PLBV46_NUM_ SLAVES-1]	Slave	I		Slave write data acknowledge
P73	Bus_Error_Det	System	O	0	Bus Error Interrupt
Interrupt					
P74	Sl_MIRQ[0:C_PLBV46_NUM_ SLAVES*C_PLBV46_NUM_ MASTERS-1]	Slave	I		Master Interrupt Request (one per master at each slave). Gives a slave the ability to indicate that it has encountered an event it deems important to the master
P75	PLB_MIRQ[0:C_PLBV46_NUM_ MASTERS-1]	Slave	O	0	Master Interrupt Request. For each master, indicates whether any slave has encountered an event that it deems important to the master
System Signals					
P76	PLB_Clk	System	I		System clock
P77	SYS_Rst	System	I		External system reset
P78	PLB_Rst	System	O	0	Registered reset output from arbitration logic
P79	SPLB_Rst[0:C_PLBV46_NUM_SLAVES-1]	System	O	0	Registered reset output from arbitration logic for system slave devices
P80	MPLB_Rst[0:C_PLBV46_NUM_ MASTERS-1]	System	O	0	Registered reset output from arbitration logic for system master devices
IBM PLB Toolkit Support[1]					
P81	PLB_SaddrAck	Simulation	O	0	Output of slave Sl_addrAck OR gate
P82	PLB_Swait	Simulation	O	0	Output of slave Sl_wait OR gate
P83	PLB_Srearbitrate	Simulation	O	0	Output of slave Sl_rearbitrate OR gate
P84	PLB_SwrDAck	Simulation	O	0	Output of slave Sl_wrDAck OR gate
P85	PLB_SwrComp	Simulation	O	0	Output of slave Sl_wrComp OR gate

Table 3: **PLB Pin Descriptions *(Cont'd)***

Port	Signal Name	Interface	I/O	Init State	Description
P86	PLB_SwrBTerm	Simulation	O	0	Output of slave Sl_wrBTerm OR gate
P87	PLB_SrdDBus[0:C_PLBV46_ DWIDTH-1]	Simulation	O	0	Output of slave Sl_rdDBus OR gate
P88	PLB_SrdWdAddr[0:3]	Simulation	O	0	Output of slave Sl_rdWdAddr OR gate
P89	PLB_SrdDAck	Simulation	O	0	Output of slave Sl_rdDAck OR gate
P90	PLB_SrdComp	Simulation	O	0	Output of slave Sl_rdComp OR gate
P91	PLB_SrdBTerm	Simulation	O	0	Output of slave Sl_rdBTerm OR gate
P92	PLB_SMBusy[0:C_PLBV46_ NUM_MASTERS-1]	Simulation	O	0	Output of slave Sl_MBusy OR gate
P93	PLB_SMRdErr[0:C_PLBV46_ NUM_MASTERS-1]	Simulation	O	0	Output of slave Sl_MRdErr OR gate
P94	PLB_SMWrErr[0:C_PLBV46_ NUM_MASTERS-1]	Simulation	O	0	Output of slave Sl_MWrErr OR gate
P95	PLB_Sssize[0:1]	Simulation	O	0	Output of slave Sl_SSize OR gate
Upper Address Extension					
P97	M_UABus(0:C_PLBV46_NUM_MASTERS*32-1)[2]	Master	I		Master upper address bits. (Only the rightmost C_PLBV46_AWIDTH-32 bits in each 32-bit field are used)
P98	PLB_UABus(0:31)[2]	Slave	O		Slave upper address bits. (Only the rightmost C_PLBV46_AWIDTH-32 bits are used)

Notes:

1. The outputs in this section are required to connect with the PLB Monitor Bus Functional Model (BFM) supplied with the IBM PLB Toolkit. These outputs are not needed otherwise, but can be used as debug signals if desired.
2. UABus ports are required for connection in the EDK tool, but the signal usage is ignored in the core. No address bits beyond 32-bits are supported at this time.

Parameter/Port Dependencies

The width of many of the PLB signals depends on the number of PLB masters and number of PLB slaves in the design. In addition, when certain features are parameterized away, the related input signals are unconnected and the related output signals are set to constant values. The dependencies between the PLB design parameters and I/O signals are shown in Table 4.

Table 4: **Parameter-Port Dependencies**

Generic or Port	Name	Affects	Depends	Relationship Description
Design Parameters				
G1	C_PLBV46_NUM_ MASTERS	P12-P39P59, P60 P61, P74, P75, P94		The width of many buses is set by the number of PLB masters in the design.
G2	C_PLBV46_NUM_ SLAVES	P58 - P72		The width of many buses is set by the number of PLB slaves in the design.
G3	C_PLBV46_AWIDTH	P13, P41		
G4	C_PLBV46_DWIDTH	P14, P26, P33, P42, P56, P60, P65, P74		
G5	C_DCR_INTFCE	G6 - G9, G15, P1-P6		If C_P2P=1, then DCR interface is not available.
G6	C_BASEADDR		G5	Unconnected if C_DCR_INTFCE=0 or C_P2P=1.
G7	C_HIGHADDR		G5	Unconnected if C_DCR_INTFCE=0 or C_P2P=1.
G8	C_DCR_AWIDTH	P1	G5	Unconnected if C_DCR_INTFCE=0 or C_P2P=1.
G9	C_DCR_DWIDTH	P4, P6	G5	Unconnected if C_DCR_INTFCE=0 or C_P2P=1.
G10	C_IRQ_ACTIVE	P73		
G11	C_EXT_RESET_ HIGH			
G12	C_PLBV46_MID_ WIDTH	P46	G1	Master ID width is log2(C_PLBV46_NUM_MASTERS).
G13	C_FAMILY			
G15	C_P2P	G5-G9, P1-P6		The DCR interface is available only in shared bus configuration. The DCR parameters have no impact and the DCR ports are unconnected if C_P2P = 1.
I/O Signals				
P1	DCR_ABus[0:C_DCR_ AWIDTH-1]		G5, G8	Width varies with the size of the DCR address bus. This input is unconnected if C_DCR_INTFCE=0 or C_P2P = 1.
P2	DCR_Read		G5	This input is unconnected if C_DCR_INTFCE=0 or C_P2P = 1.
P3	DCR_Write		G5	This input is unconnected if C_DCR_INTFCE=0 or C_P2P = 1.

Table 4: **Parameter-Port Dependencies *(Cont'd)***

Generic or Port	Name	Affects	Depends	Relationship Description
P4	DCR_DBus[0:C_DCR_ DWIDTH-1]		G5, G9	Width varies with the size of the DCR data bus. This input is unconnected if C_DCR_INTFCE=0 or C_P2P = 1.
P5	PLB_dcrAck		G5	This output is grounded if C_DCR_INTFCE=0 or C_P2P = 1.
P6	PLB_dcrDBus[0:C_DCR_ DWIDTH-1]		G5, G9	Width varies with the size of the DCR data bus. This output is grounded if C_DCR_INTFCE=0 or C_P2P = 1.
P7	PLB_rdPendPri[0:1]			
P8	PLB_wrPendPri[0:1]			
P9	PLB_rdPendReq			
P10	PLB_wrPendReq			
P11	PLB_reqPri[0:1]			
P12	M_abort[0:C_PLBV46_N UM_ MASTERS-1]		G1	Width varies with the number of PLB masters.
P13	M_ABus[0:C_PLBV46_NU M_ MASTERS*32-1]		G1, G3	Width varies with the size of the PLB address bus and the number of PLB masters.
P14	M_BE[0:C_PLBV46_NUM _ MASTERS*C_PLBV46_ DWIDTH/8-1]		G1,G4	Width varies with the size of the PLB data bus and the number of PLB masters.
P15	M_busLock[0:C_PLBV46_ NUM_MASTERS-1]		G1	Width varies with the number of PLB masters.
P16	M_TAttribute[0:16*C_PLB V46_ NUM_MASTERS-1]		G1	Width varies with the number of PLB masters.
P17	M_lockErr[0:C_PLBV46_N UM_MASTERS-1]		G1	Width varies with the number of PLB masters.
P18	M_mSize[0:C_PLBV46_N UM_ MASTERS*2 -1]		G1	Width varies with the number of PLB masters.
P19	M_priority[0:C_PLBV46_N UM_MASTERS*2 -1]		G1, G17	Width varies with the number of PLB masters. When C_ARB_TYPE = 1, the M_priority bits are ignored.
P20	M_rdBurst[0:C_PLBV46_N UM_MASTERS-1]		G1	Width varies with the number of PLB masters.
P21	M_request[0:C_PLBV46_ NUM_MASTERS-1]		G1	Width varies with the number of PLB masters.
P22	M_RNW[0:C_PLBV46_NU M_ MASTERS-1]		G1	Width varies with the number of PLB masters.
P23	M_size[0:C_PLBV46_NU M_ MASTERS*4 -1]		G1	Width varies with the number of PLB masters.
P24	M_type[0:C_PLBV46_NU M_ MASTERS*3-1]		G1	Width varies with the number of PLB masters.

Table 4: **Parameter-Port Dependencies *(Cont'd)***

Generic or Port	Name	Affects	Depends	Relationship Description
P25	M_wrBurst[0:C_PLBV46_ NUM_MASTERS-1]		G1	Width varies with the number of PLB masters.
P26	M_wrDBus[0:C_PLBV46_ NUM_MASTERS*C_PLBV 46_ DWIDTH -1]		G1, G4	Width varies with the size of the PLB data bus and the number of PLB masters.
P27	PLB_MAddrAck[0:C_PLB V46_ NUM_MASTERS-1]		G1	Width varies with the number of PLB masters.
P28	PLB_MBusy[0:C_PLBV46 _NUM_MASTERS-1]		G1	Width varies with the number of PLB masters.
P29	PLB_MRdErr[0:C_PLBV46 _ NUM_MASTERS-1]		G1	Width varies with the number of PLB masters.
P30	PLB_MWrErr[0:C_PLBV46 _ NUM_MASTERS-1]		G1	Width varies with the number of PLB masters.
P31	PLB_MRdBTerm[0:C_PLB V46_NUM_MASTERS-1]		G1	Width varies with the number of PLB masters.
P32	PLB_MRdDAck[0:C_PLB V46_ NUM_MASTERS-1]		G1	Width varies with the number of PLB masters.
P33	PLB_MRdDBus[0:C_PLB V46_NUM_MASTERS*C_ PLBV46_ DWIDTH -1]		G1, G4	Width varies with the size of the PLB data bus and the number of PLB masters.
P34	PLB_MRdWdAddr[0:C_PL BV46_NUM_MASTERS*4 -1]		G1	Width varies with the number of PLB masters.
P35	PLB_MRearbitrate[0:C_PL BV46_NUM_MASTERS-1]		G1	Width varies with the number of PLB masters.
P36	PLB_MSSize[0:C_PLBV46 _ NUM_MASTERS*2 -1]		G1	Width varies with the number of PLB masters.
P37	PLB_MWrBTerm[0:C_PLB V46_NUM_MASTERS-1]		G1	Width varies with the number of PLB masters.
P38	PLB_MWrDAck[0:C_PLBV 46_ NUM_MASTERS-1]		G1	Width varies with the number of PLB masters.
P39	PLB_MTimeout[0:C_PLBV 46_ NUM_MASTERS-1]		G1	Width varies with the number of PLB masters.
P40	PLB_abort			
P41	PLB_ABus[0:31]		G3	Width varies with the size of the PLB address bus.
P42	PLB_BE[0:C_PLBV46_D WIDTH/8-1]		G4	Width varies with the size of the PLB data bus.
P43	PLB_busLock			
P44	PLB_TAttribute[0:15]			
P45	PLB_lockErr			
P46	PLB_masterID[0:C_PLBV 46_MID_ WIDTH-1]		G12	Width varies with the number of PLB masters.
P47	PLB_MSize[0:1]			

Table 4: **Parameter-Port Dependencies *(Cont'd)***

Generic or Port	Name	Affects	Depends	Relationship Description
P48	PLB_PAValid			
P49	PLB_rdBurst			
P50	PLB_rdPrim[0:C_PLBV46_NUM_ SLAVES-1][0:C_PLBV46_NUM_SLAVES-1]			Width varies with the number of PLB slaves.
P51	PLB_RNW			
P52	PLB_SAValid			
P53	PLB_size[0:3]			
P54	PLB_type[0:2]			
P55	PLB_wrBurst			
P56	PLB_wrDBus[0:C_PLBV46_ DWIDTH-1]		G4	Width varies with the size of the PLB data bus.
P57	PLB_wrPrim[0:C_PLBV46_NUM_ SLAVES-1][0:C_PLBV46_NUM_SLAVES-1]			Width varies with the number of PLB slaves.
P58	Sl_addrAck[0:C_PLBV46_NUM_ SLAVES-1]		G2	Width varies with the number of PLB slaves.
P59	Sl_MRdErr[0:C_PLBV46_NUM_SLAVES*C_PLBV46_NUM_ MASTERS-1]		G1, G2	Width varies with the number of PLB slaves and the number of PLB masters.
P60	Sl_MWrErr[0:C_PLBV46_NUM_SLAVES*C_PLBV46_NUM_ MASTERS-1]		G1, G2	Width varies with the number of PLB slaves and the number of PLB masters.
P61	Sl_MBusy[0:C_PLBV46_NUM_SLAVES*C_PLBV46_NUM_ MASTERS-1]		G1, G2	Width varies with the number of PLB slaves and the number of PLB masters.
P62	Sl_rdBTerm[0:C_PLBV46_NUM_ SLAVES-1]		G2	Width varies with the number of PLB slaves.
P63	Sl_rdComp[0:C_PLBV46_NUM_ SLAVES-1]		G2	Width varies with the number of PLB slaves.
P64	Sl_rdDAck[0:C_PLBV46_NUM_ SLAVES-1]		G2	Width varies with the number of PLB slaves.
P65	Sl_rdDBus[0:C_PLBV46_NUM_SLAVES*C_PLBV46_DWIDTH-1]		G2,G3	Width varies with the number of PLB slaves and the size of the PLB data bus.
P66	Sl_rdWdAddr[0:C_PLBV46_NUM_ SLAVES*4-1]		G2	Width varies with the number of PLB slaves.
P67	Sl_rearbitrate[0:C_PLBV46_NUM_ SLAVES-1]		G2	Width varies with the number of PLB slaves.

Table 4: **Parameter-Port Dependencies *(Cont'd)***

Generic or Port	Name	Affects	Depends	Relationship Description
P68	Sl_SSize[0:C_PLBV46_NUM_ SLAVES*2-1]		G2	Width varies with the number of PLB slaves.
P69	Sl_wait[0:C_PLBV46_NUM_ SLAVES-1]		G2	Width varies with the number of PLB slaves.
P70	Sl_wrBTerm[0:C_PLBV46_NUM_ SLAVES-1]		G2	Width varies with the number of PLB slaves.
P71	Sl_wrComp[0:C_PLBV46_NUM_ SLAVES-1]		G2	Width varies with the number of PLB slaves.
P72	Sl_wrDAck[0:C_PLBV46_NUM_ SLAVES-1]		G2	Width varies with the number of PLB slaves.
P73	Bus_Error_Det		G5,G10	C_IRQ_ACTIVE determines the active state of the interrupt. If C_DCR_INTFCE=0, then interrupts are always enabled, otherwise, interrupts are enabled by writing to the PLB Control Register.
P74	Sl_MIRQ[0:C_PLBV46_NUM_ SLAVES*C_PLBV46_NUM_ MASTERS-1]		G1, G2	Width varies with the number of PLB slaves and the number of PLB masters.
P75	PLB_MIRQ[0:C_PLBV46_NUM_ MASTERS-1]		G1	Width varies with the number of PLB masters.
P76	PLB_Clk			
P77	SYS_Rst			
P78	PLB_Rst			
P79	SPLB_Rst[0:C_PLBV46_NUM_SLAVES-1]		G2	Width varies with the number of PLB slave devices.
P80	MPLB_Rst[0:C_PLBV46_NUM_MASTERS-1]		G1	Width varies with the number of PLB master devices.
P81	PLB_SaddrAck			
P82	PLB_Swait			
P83	PLB_Srearbitrate			
P84	PLB_SwrDAck			
P85	PLB_SwrComp			
P86	PLB_SwrBTerm			
P87	PLB_SrdDBus[0:C_PLBV46_ DWIDTH-1]		G4	Width varies with the size of the PLB data bus.
P88	PLB_SrdWdAddr[0:3]			
P89	PLB_SrdDAck			
P90	PLB_SrdComp			
P91	PLB_SrdBTerm			
P92	PLB_SMBusy[0:C_PLBV46_ NUM_MASTERS-1]		G1	Width varies with the number of PLB masters.

Table 4: **Parameter-Port Dependencies *(Cont'd)***

Generic or Port	Name	Affects	Depends	Relationship Description
P93	PLB_SMRdErr[0:C_PLBV46_ NUM_MASTERS-1]		G1	Width varies with the number of PLB masters.
P94	PLB_SMWrErr[0:C_PLBV46_ NUM_MASTERS-1]		G1	Width varies with the number of PLB masters.
P95	PLB_Sssize[0:1]		G1	Width varies with the number of PLB masters.

PLB Registers

The PLB, when configured as a shared bus (C_P2P = 0), may optionally contain DCR-accessible registers to provide error address and status information for attempted transactions that did not get a response from any slave. These registers are not available when configured as a point-to-point bus. In what follows, the term *error* refers to such a missing response, which is detected by the time out mechanism of the arbiter. If the design has been parameterized to contain a DCR interface (C_DCR_INTFCE = 1), the registers shown in Table 5 are present.

Note: The base address for these registers is set in the parameter C_BASEADDR.

Table 5: PLB DCR Registers (available only in shared bus configuration)

Register Name	Description	DCR Address	Access
PESR_MERR_DETECT	Master Error Detect Bits	C_BASEADDR + 0x00	Read/Write
PESR_MDRIVE_PEAR	Master Driving PEAR	C_BASEADDR + 0x01	Read
PESR_RNW_ERR	Read/Write Error	C_BASEADDR + 0x02	Read
PESR_LCK_ERR	Lock Error Bit	C_BASEADDR + 0x03	Read
PEAR_ADDR	PLB Error Address	C_BASEADDR + 0x04	Read(1)
PEAR_BYTE_EN	PLB Error Byte Enables	C_BASEADDR + 0x05	
PEAR_SIZE_TYPE	PLB Size and Type	C_BASEADDR + 0x06	
PACR	PLB Control Register	C_BASEADDR + 0x07	Read/Write

Notes:

1. These registers can be written if the Test Enable bit is asserted in the PACR.

PLB Error Status Registers

There are four PESR registers that provide error information - was an error detected, which Master's errant address and byte enables are in the PEARs, was the error due to a read or write transaction, and did the master lock the error condition. If a read of the PESR_MERR_DETECT register returns all zeros, then no masters detected any errors and no further reads are necessary.

PESR_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 experienced an error, then bit 0 is set. Writing a 1 to a bit in this register clears this bit and the corresponding bit in the other PESRs (PESR_MDRIVE_PEAR, PESR_RNW_ERR, and PESR_LCK_ERR).

If a particular master experienced an error and had locked the PEARs[1], writing a 1 to the corresponding bit in this register would clear and unlock the error fields of the master and unlock the PEARs. The bits in this register are reset when a 1 has been written to the register, SYS_Rst has been asserted, or a 1 has been written to the Software Reset bit in the PACR. Figure 7 shows the bit definitions of this register when the number of PLB masters is 8.

The bit definitions for PESR_MERR_DETECT are shown in Table 6. The bits is this register are reset by writing a 1 to the bit.

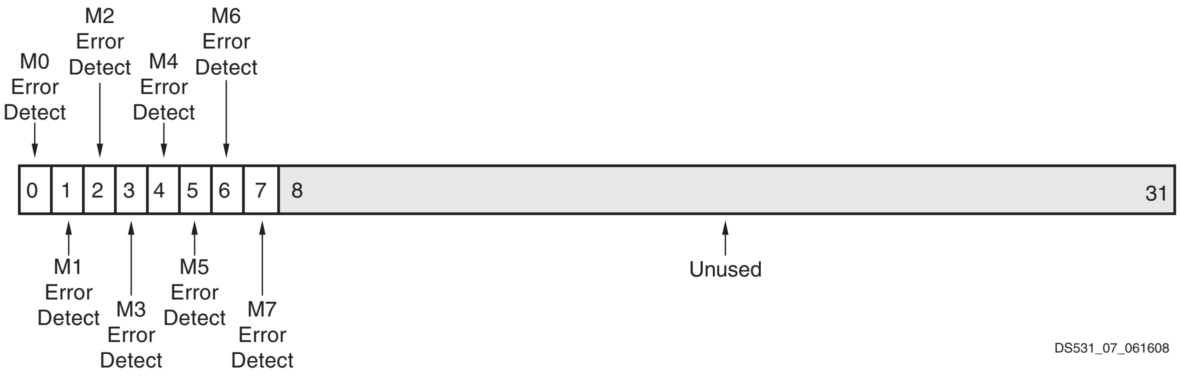


Figure 7: **PESR_MERR_DETECT (C_PLBV46_NUM_MASTERS=8, C_DCR_DWIDTH=32)**

Table 6: **PLB PESR_MERR_DETECT Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0 toC_PLBV46_NUM_ MASTERS-1	Master Error Detect	Read/Write	0	Master Error Detect. **Read**: Error detect bit for PLB Masters 0 to C_PLBV46_NUM_ MASTERS-1 respectively. 1 - error detected 0 - no error detected **Write**: Clear error bit for PLB Masters 0 to C_PLBV46_NUM_ MASTERS -1 respectively. 1 - clear and unlock corresponding master's error fields and PESRs 0 - do not clear error
Others	Unused, read as zero			

PESR_MDRIVE_PEAR: Master Driving PEAR

This register indicates which PLB Master is driving the PEARs. Each bit location in this register corresponds to a PLB Master. For example, if PLB Master 0 is driving the PEARs, then bit 0 is set. Only one master can drive the PEARs, therefore, only one bit is 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 PESR_MERR_DETECT, SYS_Rst has been asserted, or a 1 has been written to the Software Reset bit in the PACR. Figure 8 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.

[1] A master specifies whether errors are to be locked on a transaction-by-transaction basis by asserting the M_lockErr qualifier signal.

The bit definitions for PESR_MDRIVE_PEAR are shown in Table Notes:.

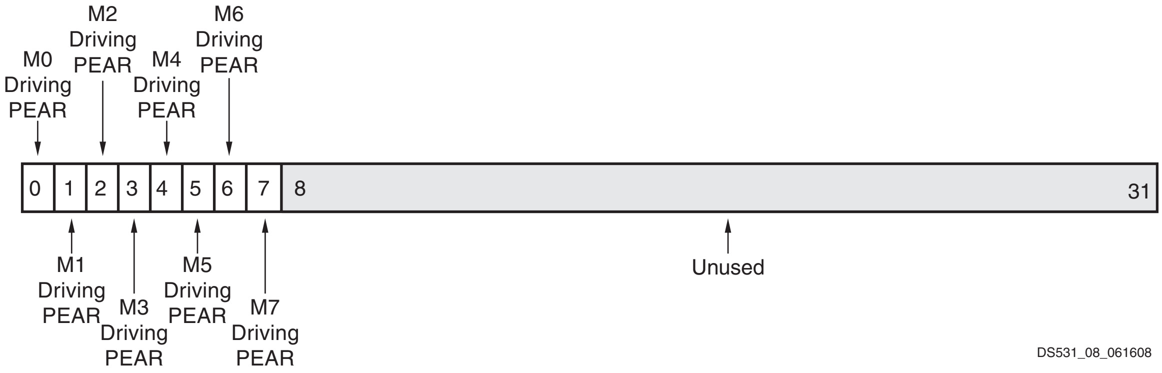


Figure 8: **PESR_MDRIVE_PEAR (C_PLBV46_NUM_MASTERS=8, C_DCR_DWIDTH=32)**

Notes: PLB PESR_MDRIVE_PEAR Bit Definitions

Bit(s)	Name	Core Access	Reset Value	Description
0 to C_PLBV46_NUM_ MASTERS-1	Master Driving PEAR	Read	0	Master Driving PEAR. **Read**: PEAR bit for PLB Masters 0 to C_PLBV46_NUM_ MASTERS-1 respectively. 1 - master is driving PEAR 0 - master is not driving PEAR **Write**: No effect.
Others	Unused, read as zero			

PESR_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 experienced an error during a read operation, bit 0 would be set.

If PLB Master 1 experienced 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 PESR_MERR_DETECT, SYS_Rst has been asserted, or a 1 has been written to the Software Reset bit in the PACR. Figure 9 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.

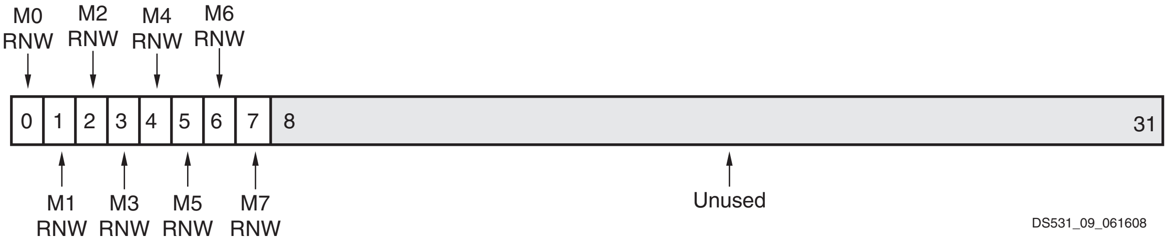


Figure 9: **PESR_RNW_ERR (C_PLBV46_NUM_MASTERS=8, C_DCR_DWIDTH=32)**

The bit definitions for PESR_RNW_ERR are shown in Table 7.

Table 7: **PLB PESR_RNW_ERR Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0 to C_PLBV46_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.
Others	Unused, read as zero			

PESR_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 error fields of the master are locked, which means that 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 PESR_MERR_DETECT, SYS_Rst has been asserted, or a 1 has been written to the Software Reset bit in the PACR. Figure 10 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. The bit definitions for PESR_LCK_ERR are shown in Table 8.

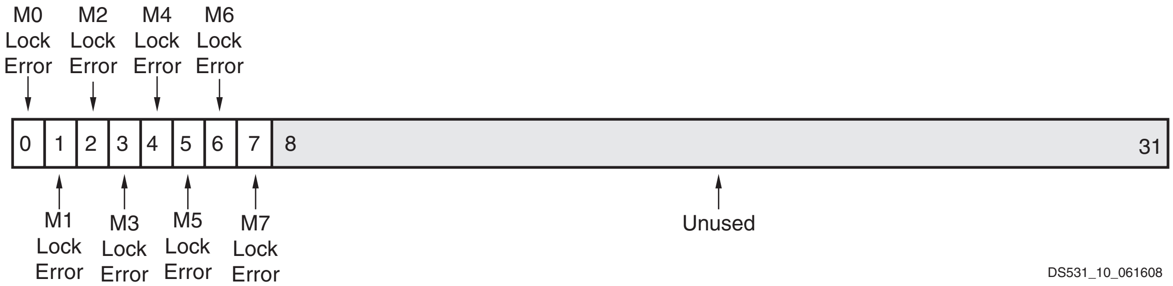


Figure 10: **PESR_LCK_ERR (C_PLBV46_NUM_MASTERS=8, C_DCR_DWIDTH=32)**

Table 8: PLB PESR_LCK_ERR Bit Definitions

Bit(s)	Name	Core Access	Reset Value	Description
0 to C_PLBV46_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.
Others	Unused, read as zero			

Bus Error Address Registers

There are three PEAR registers. These registers contain the PLB address, PLB byte enables, PLB size, and PLB type of the transaction that caused the error.

PEAR_ADDR: PLB Error Address register

This register contains the low-order 32 bits of the PLB address of the transaction that caused the error as shown Figure 11. This register is cleared when SYS_Rst is asserted or a 1 is written to the Software Reset bit.

The bit definitions for PEAR_ADDR are shown in Table 9.

Table 9: **PLB PESR_LCK_ERR Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0 to 31	Bus Error Address Low	Read/Write	0	Bus Error Address, low bits **Read**: PLB address where error occurred. **Write**: If the Test Enable bit is asserted in the PACR, this field is writable. Otherwise, a write has no effect.

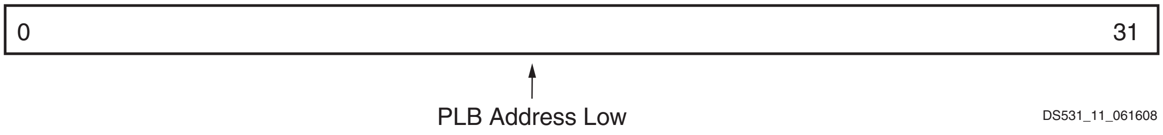


Figure 11: **PEAR_ADDR Register**

PEAR_BYTE_EN: PLB Error Byte Enables and High-order Address

This register contains on the left the values of the PLB byte enables and on the right the high-order address bits of the transaction that caused the error as shown in Figure 12. The width of the PLB byte enable bus is the width of the PLB data bus divided by 8. Therefore, if the PLB data bus is 128 bits wide, there are 16 byte enables. This register is cleared when SYS_Rst is asserted or a 1 is written to the Software Reset bit.

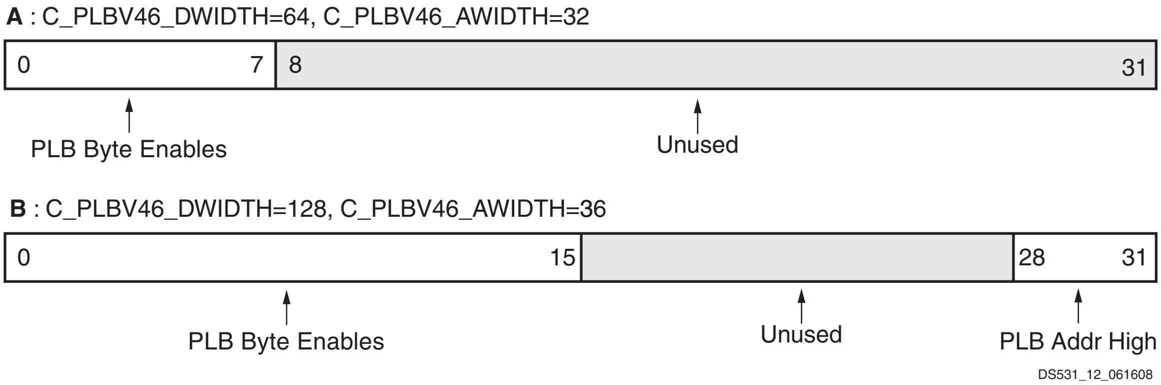

A : C_PLBV46_DWIDTH=64, C_PLBV46_AWIDTH=32
0
7
8
31
PLB Byte Enables
Unused
B : C_PLBV46_DWIDTH=128, C_PLBV46_AWIDTH=36
0
15
28
31
PLB Byte Enables
Unused
PLB Addr High
DS531_12_061608

Figure 12: **PEAR_BYTE_EN Register**

The bit definitions for PEAR_BYTE_EN are shown in Table 10.

Table 10: **PLB PEAR_BYTE_EN Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0 to C_PLBV46_DWIDTH/8-1	Bus Error Address High	Read Write(1)	0	Bus Error Byte Enables. **Read**: PLB byte enable value when error occurred **Write**: If the Test Enable bit is asserted in the PACR, this field is writable.Otherwise, a write has no effect.
64-C_PLBV46_AWIDTH to 31				Bus Error Byte Enables, high bits.
Others	Unused, read as zero			

Notes:

1. This register can be written if the Test Enable bit is asserted in the PACR. Unused bits are not writable.

PEAR_SIZE_TYPE: PLB Error Size and Type

This register contains the values of the PLB size and type during the transaction that caused the error as shown in Figure 13. This register is cleared when SYS_Rst is asserted or a 1 is written to the Software Reset bit. The bit definitions for PEAR_SIZE_TYPE are shown in Table 11.

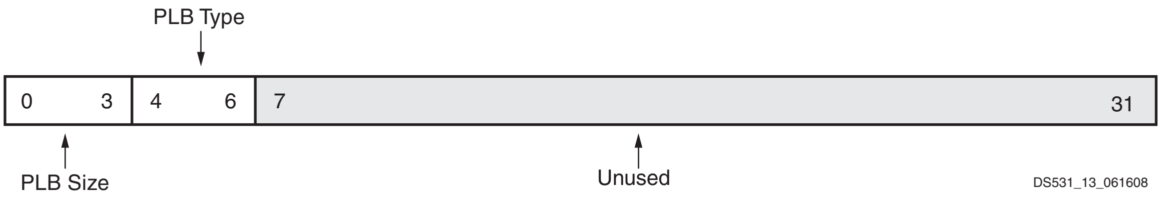


Figure 13: **PEAR_SIZE_TYPE Register**

Table 11: **PLB PEAR_SIZE_TYPE Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0 to 3	PLB Size	Read Write(1)	’0000’	PLB Size. **Read**: PLB size value when error occurred **Write**: If the if the Test Enable bit is asserted in the PACR, this field is writable.Otherwise, a write has no effect.
4 to 6	PLB Type	Read Write(1)	’00’	PLB Type. **Read**: PLB type value when error occurred **Write**: If the if the Test Enable bit is asserted in the PACR, this field is writable.Otherwise, a write has no effect.
Others	Unused, read as zero			

Notes:

1. This register can be written if the Test Enable bit is asserted in the PACR. Unused bits are not writable.

PLB Control Register

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

PACR: PLB Control Register

This register contains one bit that enables or disables the interrupt request and another bit used to reset the PLB as shown in Figure 14.

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

Also note that when the Software reset bit is asserted, ALL registers and flip-flops within the PLB including all PEAR/PESR registers are reset. This reset occurs independent of the current PLB transaction state, therefore, this reset should be used carefully.

This register is reset to the default state whenever SYS_Rst is asserted or a 1 is written to the Software Reset bit. The bit definitions for PACR are shown in Table 12.

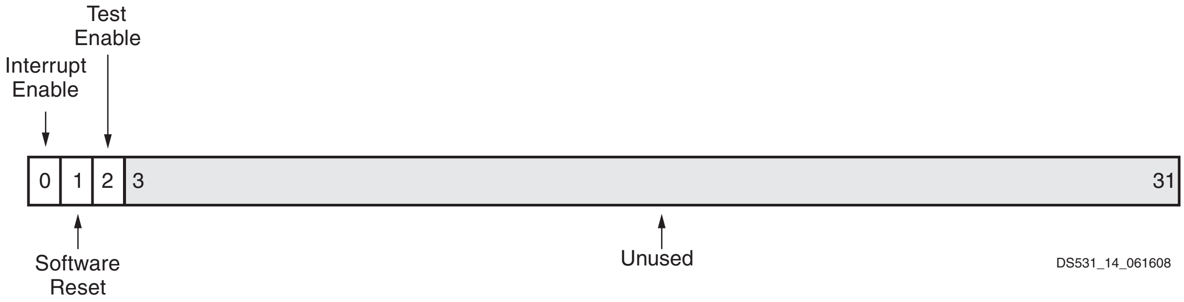


Figure 14: **PACR (C_DCR_DWIDTH=32)**

Table 12: **PLB PACR Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0	Interrupt Enable	Read/Write	1	Interrupt Enable. **Read**: PLB 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 because it is reset whenever a 1 is written to it. **Write**: 1 - reset the PLB 0 - resume normal PLB operation

Table 12: **PLB PACR Bit Definitions *(Cont'd)***

Bit(s)	Name	Core Access	Reset Value	Description
2	Test Enable	Read/Write	0	Test Enable. **Read**: Test Enable **Write**: 1 - Enable writing to the PEAR registers 0 - PEAR registers are not writable
3 to C_DCR_DWIDTH-1	Unused, read as zero			

Notes:

1. Use the software reset cautiously because the software reset will reset the entire PLB regardless of the current PLB transaction state.

PLB Interrupt Description

The PLB has one interrupt request output called Bus_Error_Det. The interrupt is signaled when a bus transfer times out because it is not responded to by any slave. 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 is generated whenever any time out error is detected regardless of whether masters have locked their error fields or not. See <RD Red>"PLB Registers" on page 19. If the parameter C_DCR_INTFCE is 0, which indicates that there is no DCR interface, interrupts will remain enabled because the default state of the Interrupt Enable bit in the PACR is asserted.

Master Interrupt Request

The Xilinx PLB supports the Sl_MIRQ (0 to C_PLBV46_NUM_ SLAVES*C_PLBV46_NUM_ MASTERS - 1) signal, which allows each slave to signal to any master that it has encountered an event that it consders important to that master. The only function of the Xilinx PLB with respect to the MIRQ signals is to OR all the Sl_MIRQ signals for a given master into the corresponding bit of PLB_MIRQ(0 to C_PLBV46_NUM_ MASTERS-1).

PLB Block Diagram

Figure 15 provides a comprehensive block diagram of the PLB.

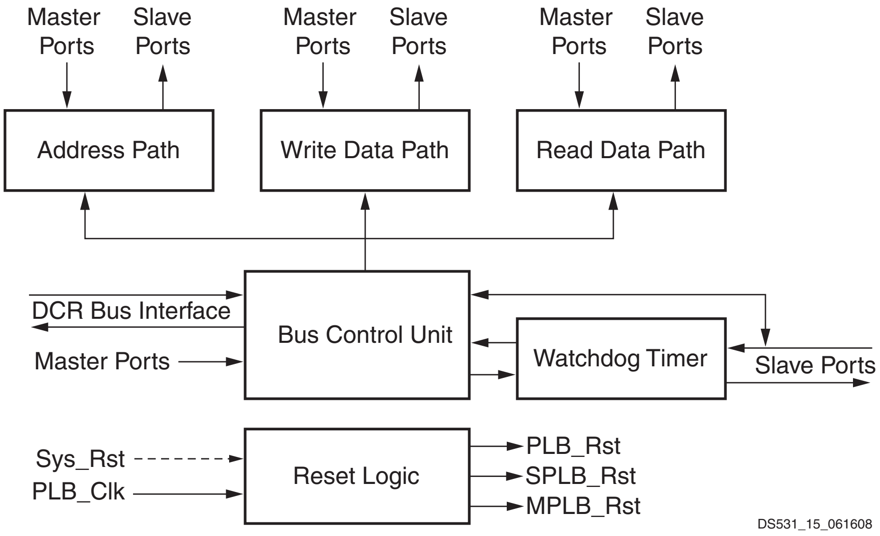


Figure 15: **PLB Block Diagram**

Address Path Unit

The PLB address path unit contains the muxing needed to select the master address which is driven to the slave devices on the PLB address output.

Write Data Path Unit

The PLB write data path unit contains the steering logic needed for the master and slave write data buses.

Read Data Path Unit

The PLB read data path unit contains the steering logic needed for the master and slave read data buses.

Bus Control Unit

The PLB bus control unit consists of a bus arbitration control unit that manages the address and data flow through the PLB and DCRs. The bus arbitration control unit supports arbitration for 16 PLB masters. The address and data flow control logic provides address pipelining and address and data steering support for 16 PLB masters and 8 PLB slaves.

DCR-accessible, PLB registers may be optioned in for use in reporting timeout errors if the bus is configured as a shared bus. The registers are accessed by using the *move from device* control register (*mfdcr*) and *move to device* control register (*mtdcr*) instructions, which move data between the device control registers and the processor's general purpose registers.

See the "PLB Registers" section for additional information.

Bus Arbitration

The PLB v4.6 arbitration type is selected using the C_ARB_TYPE design parameter. When C_ARB_TYPE = 0, the arbitration type is a fixed priority arbitration as discussed in <RD Red>"Basic Operation" on page 2.

When C_ARB_TYPE = 1, the arbitration logic is configured in a round robin scheme. The round robin implementation on the PLB v4.6 is such that the last master to win arbitration and be *granted* the bus, will be the lowest priority available master to be granted the bus in the next arbitration cycle. The arbitration cycle is indicated by either the assertion of a Sl_AddrAck, Sl_reArbitrate, or a PLB time out condition. Round robin arbitration keeps an embedded priority scheme fixed amongst the masters in the system and rotates the priority ordering based on the last master to win the bus. The arbiter is only able to arbitrate on requesting masters indicated by the M_Request signals at the arbitration clock cycle.

The round robin scheme, allows masters that may have been starved for the bus, a fair chance of being granted the bus. An example of round robin implementation with C_NUM_MASTERS = 3 is shown in Figure 16.

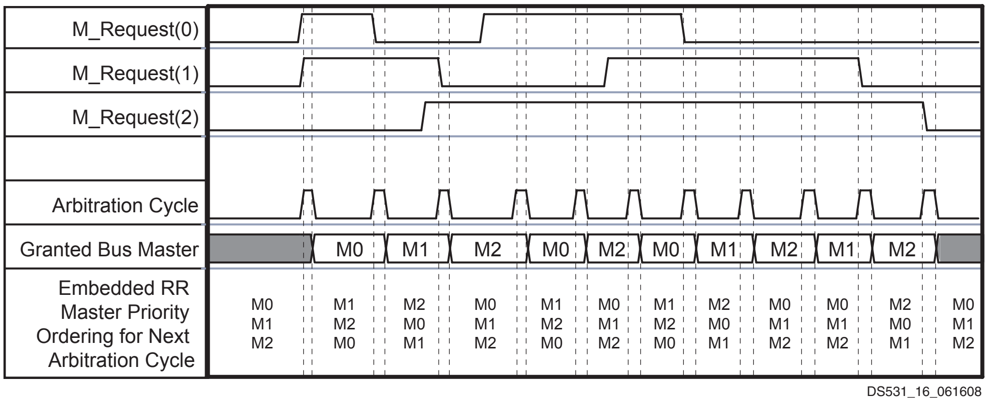


Figure 16: **Example of Round Robin Arbitration**

Watchdog Timer

The PLB watchdog timer is used to generate the PLB_MTimeout response when no slave responds. The watchdog time is set to 16 clock cycles.

Reset Logic

The PLB v4.6 does not include any power-on reset circuitry to ensure that a PLB reset is generated upon power-on if no external reset (Sys_Rst) is provided. It is the assumption in PLB v4.6 systems, that the designer will include the proc_sys_reset core to ensure a power-on reset is asserted for at least 16 clock cycles.

The reset logic in the PLB v4.6 core will provide one stage of synchronization from the external reset (Sys_Rst) to the output reset signals, PLB_Rst, SPLB_Rst, and MPLB_Rst. The PLB reset is synchronous to the PLB clock.

The additional slave and master vectorized reset signals (SPLB_Rst and MPLB_Rst), have identical timing characteristics as the PLB reset (PLB_Rst).

Point-to-Point Mode

The PLB v4.6 core can be optimized for point-to-point connections by setting the design parameter, C_P2P = 1. This configuration allows for optimization when only a single master and single slave device are required to communicate.

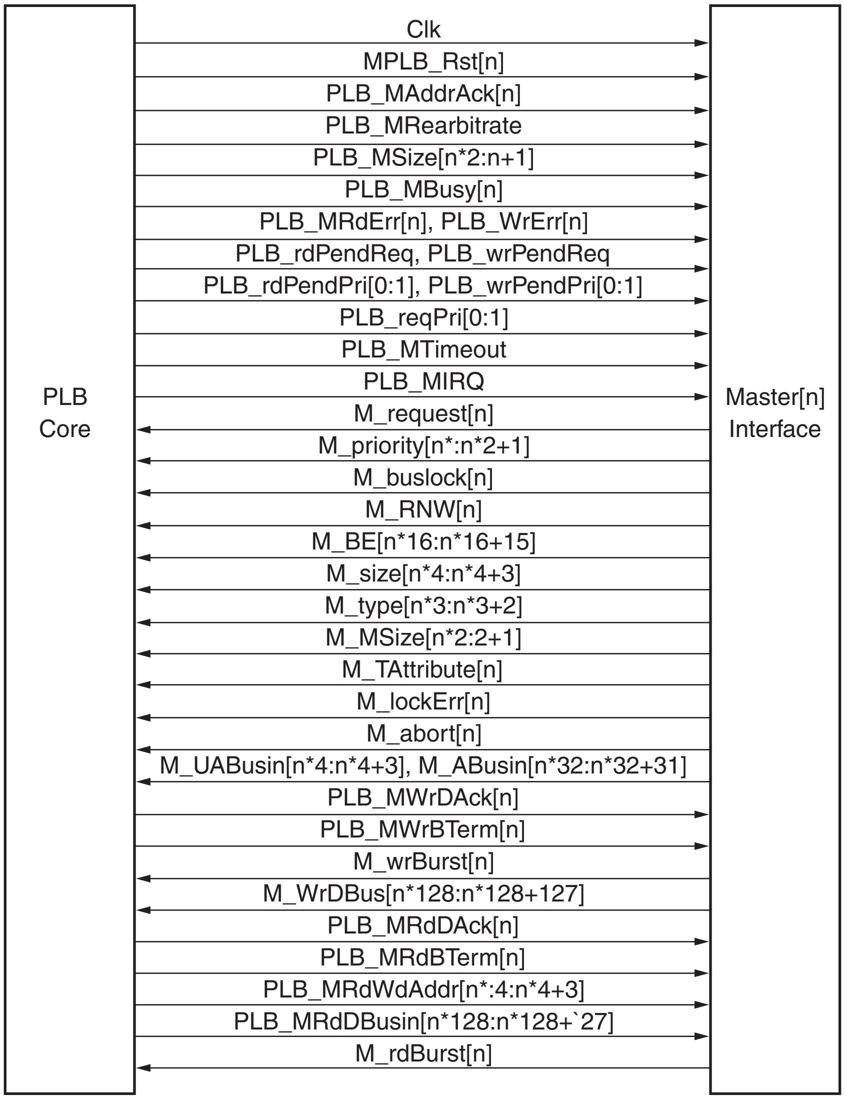
In this mode, components such as the Address Path, Write & Read Data Path Units, and Bus Control Unit are not included in the core to minimize resource utilization and improve latency. In point-to-point mode, the M_Request signal can be directly routed to PLB_PAValid with minimal latency. The point-to-point mode still incorporates the Watchdog Timer to allow PLB_MTimeout to be asserted if the slave device does not respond to PLB_PAValid.

Enabling address pipelining is configurable in point-to-point mode. By setting the design parameter, C_ADDR_PIPELINING_TYPE = 1, a 2-level pipeline is incorporated in the design. In this configuration, an arbitration state machine controls the assertion of PLB_PAValid and PLB_SAValid.

When the PLB is configured in a point-to-point mode, C_P2P = 1, the bus will ignore any assertion by the master device on the M_busLock signal. Since only one master is utilizing the bus, there is no need to drive PLB_busLock and the bus will default this output to ’0’.

Master[n] Interface

Figure 17 shows all master[n] interface I/O signals (where *n* is the number of a master 0 to C_PLBV46_NUM_ MASTERS-1). Refer to the *IBM 128-Bit Processor Local Bus Architectural Specification (v4.6)* for detailed functional descriptions of these signals. Note that C_PLBV46_DWIDTH =128 and C_PLBV46_AWIDTH=36 in this diagram.



DS531_17_061608

Figure 17: **Master[n] Interface**

Slave Interface[m]

Figure 18 demonstrates all slave[m] interface I/O signals (where m = 0 to C_PLBV46_NUM_ SLAVES-1). Refer to the *IBM 128-Bit Processor Local Bus Architectural Specification (v4.6)* for detailed functional descriptions of these signals. Note that C_PLBV46_NUM_ MASTERS=8, C_PLBV46_DWIDTH=128, and C_PLBV46_AWIDTH=36 in this diagram.

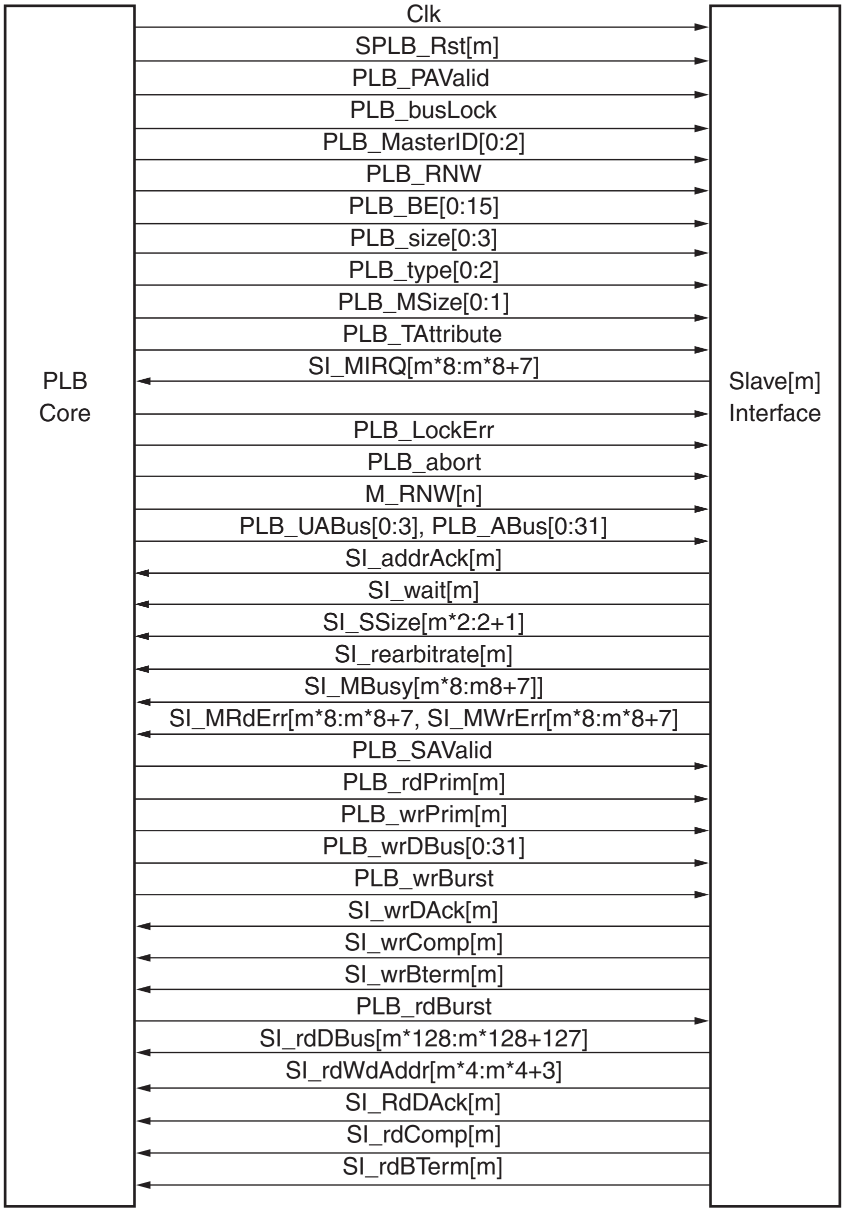


Figure 18: **Slave[n] Interface**

DCR Interface

The device control register (DCR) interface allows the CPU core in the system to read and write the DCRs in the PLB. The DCR interface is only available in shared bus configuration. For additional information on the DCR bus, see the *IBM 32-Bit Device Control Register Bus Architecture Specifications*.

Figure 19 demonstrates all DCR interface input/output signals when C_DCR_DWIDTH = 32 and C_DCR_AWIDTH=10.

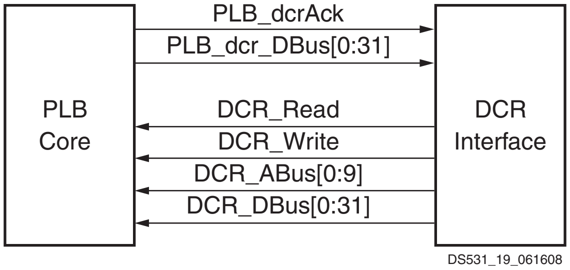


Figure 19: **DCR Interface**

PLB Operations

The *IBM 128-Bit Processor Local Bus Architectural Specification (v4.6)* document provides a comprehensive discussion on the various PLB operations and transfers. The reader is referred to that document for further protocol description and timing diagrams. Different specific timing relationships can conform to the same protocol and might reflect different tradeoffs between F_{MAX}, latency and resource usage. Some expected timing characteristics of a prospective implementation are noted below, without imposing them as rigid requirements.

- M_request to PLB_PAValid delay
  - 2 cycles when the number of masters is two or more
  - 1 (or possibly 0) cycles when there is one master
- Master-to-slave signals flow combinatorially through a multiplexer whose selection is the active master
- Slave-to-master signals flow combinatorially through an OR concentrator over all slaves, then through a demultiplexer to the active master
- PLB_rdPrim and PLB_wrPrim react combinatorially to the respective Sl_rdComp or SL_wrComp

Bus Time-Out

Because the time out concept for PLB V4.6 differs from PLB V3.4, it is illustrated here in more detail. Please note that a time out finishes a transaction in the address phase instead of proceeding, as with V3.4, to a data phase with the arbiter supplying artificial address and data acknowledges. Figure 20 shows a bus time-out for an attempted transfer to which no slave responds within the time out interval of 16 clocks. The PLB arbitration logic samples the **Sl_wait**, **Sl_rearbitrate** and **M_Abort** signals 16 cycles after the initial assertion of the **PLB_PAValid** signal, and if all are negated, it asserts the **PLB_MTimeout** signal. This completes all handshaking for the transfer.

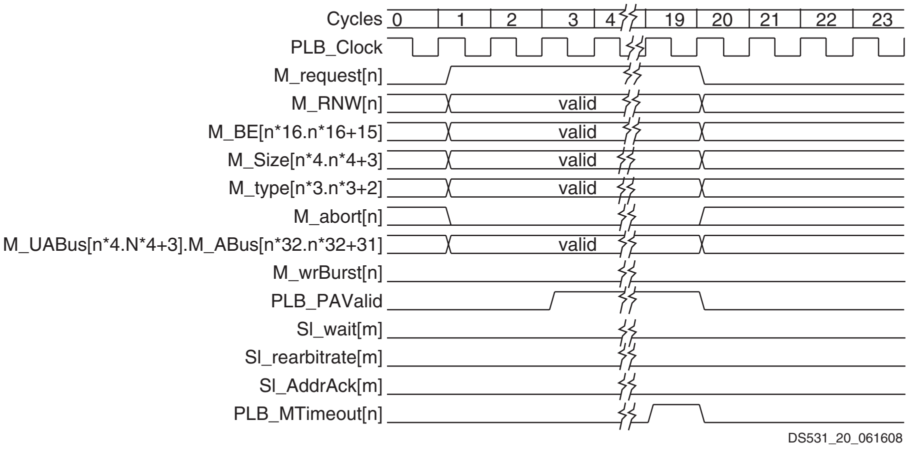


Figure 20: **Bus Time-Out**

Time-Out Suppression

If a slave cannot respond within 16 cycles, it can suppress the time out and buy more time. Figure 21 shows a slave suppressing the time out by responding with **Sl_wait[m]** within 16 clocks. When the slave is eventually ready, it responds with **Sl_AddrAck[m]** (shown) or **Sl_rearbitrate[m]**.

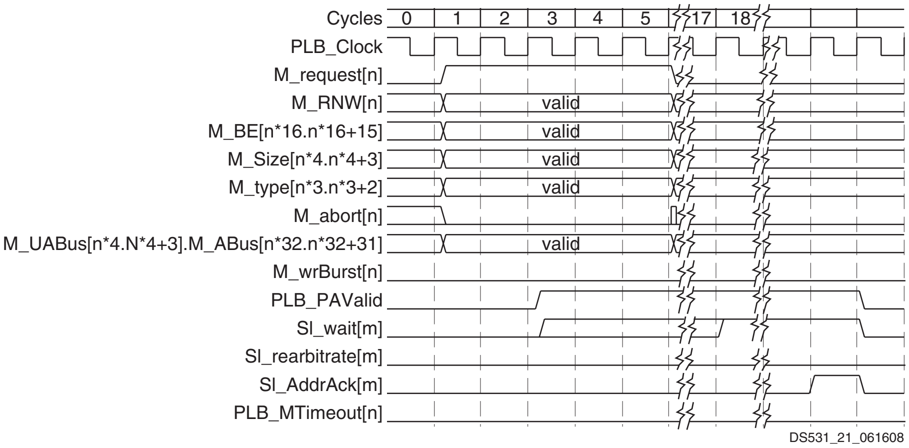


Figure 21: **Time-Out Suppression**

Design Implementation

Target Technology

The target technology is an FPGA listed in EDK Supported Device Families.

Device Utilization and Performance Benchmarks

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

The PLB benchmarks shown in Table 13 are for a Virtex®-5 (XC5VFX70T) device.

Table 13: **PLB FPGA Performance and Resource Utilization Benchmarks**(Notes:

Parameter Values					Device Resources			
C_PLBV46_NUM_ MASTERS	C_PLBV46_NUM_ SLAVES	C_P2P (default = 0)	C_ADDR_ PIPELINING_TYPE (default=1)	C_ARB_TYPE (default = 0)	Slices	Slice Registers	Slice LUTs	f_{MAX} (MHz)
1	1	1	0	N/A	4	10	14	635.3
1	1	1	1	N/A	11	17	35	500.2
1	1	0	1	0	46	128	67	378.7
1	4	0	1	0	64	131	149	357.5
4	1	0	1	0	242	175	559	220.8
4	4	0	0	0	259	156	658	252.5
4	4	0	1	0	243	179	655	222.8
4	4	0	1	1	278	197	683	227.6
8	8	0	0	0	354	184	1214	235.8
8	8	0	1	0	331	221	1185	203.2
8	16	0	1	0	438	229	1296	228.4

Notes:

1. These benchmark designs contain only the Plb with registered inputs and outputs without any additional logic. Benchmark numbers approach the performance ceiling rather that representing performance under typical user conditions.

Specification Exceptions

There are few differences that should be noted by the reader of this specification who also reads the IBM PLB v4.6 specification.

PLB Bus Structure

The Xilinx PLB provides the full PLB bus structure. No external OR gates are required for the slave input data. Each PLB slave connects directly to the Xilinx PLB as shown in <RD Red>Figure 1 on page 2.

I/O Signals

The master interface signals and many of the PLB signals have been combined into a bus with an index that varies with the number of masters. This modification more easily supports the parameterization of the number of masters and the number of slaves supported by the Xilinx PLB.

Signals that have the master designator of *Mn* in the signal name in the IBM PLB specification have a master designator of *M* in the signal name in this document. For example, the signal called **PLB_MnReabitrate** in the IBM specification is called **PLB_MRearbitrate** here. Similarly, **Mn_RNW** is called **M_RNW**.

The optional parity concept of PLB v4.6 is not supported. No parity signals are included in the ports.

Differences between the PLB v4.6 and the v3.4

The Xilinx PLB v4.6 bus logic core, the subject of this data sheet, is derived from the PLB v3.4 core. The major difference between the Xilinx PLB v4.6 and Xilinx PLB v3.4 cores are:

- Maximum data width of 128 bits versus 64 bits.
- **M_UABus** and **PLB_UABus** signals added to allow address to grow beyond 32 bits. Currently, all Xilinx soft IP and EDK tool only utilize 32-bits of the initial PLB v4.6 implementation. The UABus is included as a required port connection on peripherals, but is driven to zeros by Masters and internally ignored by Slave IP devices. The PLB v4.6 upper address bus will be included in the required port interface for peripherals and included in the bus structure multiplexing by the arbiter.
- **PLB_MTimeout** signal added. If no slave responds to the **PAValid** assertion, the arbiter asserts the **PLB_MTimeout** signal for one clock. In the PLB v3.4 version the arbiter had responsibility to complete the address acknowledge and data acknowledges (with error qualification on the data acknowledges).
- **SL_MIRQ** and **PLB_MIRQ** signals added. These allow any slave to signal an event deemed important to any master.
- **Sl_MErr** and **PLB_MErr** signals split into separate read and write versions: **SL_MWrErr**, **Sl_MRdErr**, **PLB_MWrErr** and **PLB_MRdErr**.
- **PLB_pendReq** and **PLB_pendPri** signals split into separate read and write version: **PLB_wrPendReq**, **PLB_rdPendReq**, **PLB_wrPendPri** and **PLB_rdPendPri**.
- 16-bit **M_TAttribute** and **PLB_TAttribute** signals added. The compressed, guarded and ordered qualifiers are now expressed as TAttribute values, and separate signals for these qualifiers are no longer present.

Support

Xilinx provides technical support for this LogiCORE product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled *DO NOT MODIFY*.

Reference Documents

The following documents contain reference information important to understanding the Xilinx PLB design.

1. DS400 Processor Local Bus (PLB) v3.4
2. IBM 128-Bit Processor Local Bus Architectural Specification (v4.6)
3. IBM 32-Bit Device Control Register Bus Architecture Specifications, Version 2.9

Revision History

The following table shows the revision history for this document.

Date	Version	Description of Revisions
6/13/08	1.0	Initial Xilinx release.
11/13/08	1.1	Edited "Address Pipelining" section; converted to current data sheet template.
4/24/09	1.2	Replaced references to supported device families and tool name(s) with hyperlink to PDF file.

Notice of Disclaimer

Xilinx is providing this design, code, or information (collectively, the “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.