PLBV46 Interface Simplifications

SP026 (v1.0) October 11, 2007

XILINX®

Xilinx is disclosing this Specification (hereinafter "the Specification") to you for use in the development of designs to operate on, or interface with, Xilinx FPGAs.

Xilinx does not assume any liability arising out of the application or use of the Specification; nor does Xilinx convey any license under its patents, copyrights, or any rights of others. You are responsible for obtaining any rights you may require for your use or implementation of the Specification. Xilinx reserves the right to make changes, at any time, to the Specification as deemed desirable in the sole discretion of Xilinx. Xilinx assumes no obligation to correct any errors contained herein or to advise you of any correction if such be made. Xilinx will not assume any liability for the accuracy or correctness of any engineering or technical support or assistance provided to you in connection with the Specification.

THE SPECIFICATION IS PROVIDED "AS IS" WITH ALL FAULTS, AND THE ENTIRE RISK AS TO ITS FUNCTION AND IMPLEMENTATION IS WITH YOU. YOU ACKNOWLEDGE AND AGREE THAT YOU HAVE NOT RELIED ON ANY ORAL OR WRITTEN INFORMATION OR ADVICE, WHETHER GIVEN BY XILINX, OR ITS AGENTS OR EMPLOYEES. XILINX MAKES NO OTHER WARRANTIES, WHETHER EXPRESS, IMPLIED, OR STATUTORY, REGARDING THE SPECIFICATION, INCLUDING ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, AND NONINFRINGEMENT OF THIRD-PARTY RIGHTS.

IN NO EVENT WILL XILINX BE LIABLE FOR ANY CONSEQUENTIAL, INDIRECT, EXEMPLARY, SPECIAL, OR INCIDENTAL DAMAGES, INCLUDING ANY LOST DATA AND LOST PROFITS, ARISING FROM OR RELATING TO YOUR USE OF THE SPECIFICATION, EVEN IF YOU HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THE TOTAL CUMULATIVE LIABILITY OF XILINX IN CONNECTION WITH YOUR USE OF THE SPECIFICATION, WHETHER IN CONTRACT OR TORT OR OTHERWISE, WILL IN NO EVENT EXCEED THE AMOUNT OF FEES PAID BY YOU TO XILINX HEREUNDER FOR USE OF THE SPECIFICATION. YOU ACKNOWLEDGE THAT THE FEES, IF ANY, REFLECT THE ALLOCATION OF RISK SET FORTH IN THIS AGREEMENT AND THAT XILINX WOULD NOT MAKE AVAILABLE THE SPECIFICATION TO YOU WITHOUT THESE LIMITATIONS OF LIABILITY.

The Specification is not designed or intended for use in the development of on-line control equipment in hazardous environments requiring fail-safe controls, such as in the operation of nuclear facilities, aircraft navigation or communications systems, air traffic control, life support, or weapons systems ("High-Risk Applications"). Xilinx specifically disclaims any express or implied warranties of fitness for such High-Risk Applications. You hereby acknowledge that use of the Specification in such High-Risk Applications is fully at your risk.

© 2007 Xilinx, Inc. All rights reserved. XILINX, the Xilinx logo, and other designated brands included herein are trademarks of Xilinx, Inc. All other trademarks are the property of their respective owners.

© 2007 Xilinx, Inc. All rights reserved. XILINX, the Xilinx logo, and other designated brands included herein are trademarks of Xilinx, Inc. PowerPC is a trademark of IBM, Inc. All other trademarks are the property of their respective owners.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
10/11/07	1.0	Xilinx Initial Release

Table of Contents

Section 1: Overview

Motivations 5
  Supersedes PLB v3.4 5
  Replace OPB 5
  Allow for Shared Bus and Optimized Point to Point Connection Schemes 6

Section 2: General Simplifications to PLBV4.6

Simplification Feature Subsets 7
Required Interface Signal Set 7
Mirroring /Steering 7
Addressing Restrictions 8
Transfer Type Restrictions 8
Parity 8
Transfer Attributes 8
Aborts 9
Cacheline Transfers 9
Indeterminate Burst Support 9
Fixed Length Burst Support 9
Premature Burst Terminations 9
Address Pipelining 10
Pending Request Status 10
Slave to Master Interrupts 10
Parameterization 11
  Parameter Naming Conventions 11
  Predefined Parameter Names and definitions 11
  PLBV46 Port Naming 13
  PLBV46 Interface Port Naming Convention 14

Section 3: Baseline Peripheral Requirements

Baseline Peripheral Use Case 15
Baseline Slave Feature Subset Summary 15
  Supported Functions 15
  Unsupported Functions 16
Baseline Slave PLBV46 Interface Signal Set and Usage 16
Baseline Slave Parameterization 18
Baseline Slave Interconnect Considerations 19
  Point to Point Interconnect Optimizations 19
  Shared Bus Interconnect 19
Baseline Slave Normal Transfers 19
Baseline Slave Exceptions 19
Baseline Master Feature Subset Summary 19

Supported Functions 19
Unsupported Functions 20
Baseline Master PLBV46 Interface Signal Set and Usage 20
PLBV46 Interface Signal Set 20
Additional Baseline Master Signals 22
Baseline Master Parameterization 22
Baseline Master Interconnect Requirements 22
Point to Point Interconnect 22
Shared Bus Interconnect 22
Baseline Master Normal Transfers 22
Baseline Master Exceptions 23

Section 4: Peripheral Performance Requirements

Use Case 25
Performance Slave Feature Subset 25
Supported Functions 25
Unsupported Functions 26
Optional Functions 27
Performance Slave Signal Set and Usage 27
Performance Slave Parameterization 29
Performance Slave Interconnect Requirements 30
Performance Slave Normal Transfers 30
Performance Slave Exceptions 30
Performance Master Feature Subset 31
Supported Features 31
Optional Features 31
Unsupported Features 32
Performance Master Signal Set and Usage 32
Additional Performance Master Signals 34
Performance Master Parameterization 34
Performance Master Interconnect Requirements 35
Performance Master Normal Transfers 35
Performance Master Exceptions 35

Section 5: PLBV46 Bus Core

Supported Features 37
Unsupported Features 37
PLBV46 Bus Signal Set and Usage 37
PLBV46 Bus Parameterization 37

Section 6: EDK Engines (platgen) Support for Parameters

Bus Interface 39
Parameters 39
EDK Automation of Parameter Value Assignments 40
PLBV46 to PLBV46 Bridge Considerations 40
Point to Point Interconnect Considerations 41
Appendix Usage 43
Feature Subset 43
Core 43
Typical Master and Slave Core Examples 43
IP 45
Master 45
Slave 45
Target 45
Native Data Width 45
Connection Topologies: 45
Shared Bus 46
Point to Point 46

Section 1

Overview

1.1 Motivations

1.1.1 Supersedes PLB v3.4

In order to streamline the customer-visible IP catalog as well as the peripheral IP development and maintenance efforts, all IP with PLB interfaces are to use the Xilinx-prescribed PLB v4.6 protocol, as defined in this spec. Active peripheral IP previously implemented with PLB v3.4 interfaces are to be updated to instead use PLB v4.6 without feature regression. It is further planned that the PPC405 processor block IP (wrapper) in Virtex4 is to be updated to use PLB v4.6 interfaces instead of PLB v3.4.

1.1.2 Replace OPB

In order to simplify the customer-visible IP catalog and associated system design process, the OPB bus protocol is to be phased out over time. Active IP previously implemented with OPB interfaces are to be updated to instead use PLB v4.6. It is further planned that the MicroBlaze processor will be updated to use a PLBv46 interface instead of OPB for peripheral and non-cached memory access. As a general guideline, such IP should not significantly grow in resource utilization when moving from OPB to the corresponding subset of PLB4.6, assuming IP capabilities remain the same. Over time, the simplification motivation may encompass and eliminate customer visibility of other interconnect protocols currently in use in EDK.

1.1.3 Allow for Shared Bus and Optimized Point to Point Connection Schemes

PLBV4.6 is generally viewed as a shared bus topology that incorporates a centralized bus/arbiter with multiple masters and multiple slaves attached to it. Xilinx recognizes that modern SOC designs are moving away from this type of topology and instead embracing point to point interconnection schemes for high performance designs. Xilinx's PLBV46 solution is designed to support both types of topologies within a SOC design and therefore allow the User to exploit the advantages of both based on the system needs. This is accomplished via XPS tool enhancements as well as employing configurable optimizations in the PLBV46 Bus/Arbiter building block and PLBV46 Slave Interfaces.

General Simplifications to PLBV4.6

The following sections define the global Xilinx simplifications to the PLB V4.6 usage in order to minimize resource utilization and complexity of Xilinx IP connected via PLB V4.6. Even though not all of the available PLB V4.6 features are being used, PLB protocol requirements are being obeyed except as identified in Section Known PLB V4.6 Transfer Protocol Exceptions. This facilitates the use of existing PLB tools (such as the IBM BFM protocol monitor) that can be attached and used in system simulations with minimum or no reported protocol violations.

2.1 Simplification Feature Subsets

Two feature subsets are defined as part of the Xilinx PLBV46 interface simplifications. The first is identified as the **Baseline** Feature subset. Baseline PLBV46 interfaces are targeted for peripherals requiring low performance and minimal resource utilization. Baseline interface peripherals are limited to single data beat transfers of 1 to 4 bytes (32-bits). The second feature subset is targeted towards higher performance requirements and is therefore identified as the **Performance** Feature subset. The Performance subset incorporates wider transfer widths (32, 64, or 128 bits) and multi-data beat transfers in the form of Fixed Length Bursts and Cachelines. However, this will be at the expense of increased complexity and higher resource utilization within the peripherals interface design.

2.2 Required Interface Signal Set

Feature Subset Applicability: Baseline and Performance

The required PLBV46 interface signal set for all attaching IP is the same regardless of the Xilinx Feature subset supported. It is as defined in the IBM PLB V4.6 Spec with the optional Parity signaling omitted. Xilinx simplifications to the PLB V4.6 features are carried out via the implementation of the Master and Slave designs. Included with the feature subset definitions described in this document are the requirements for which PLBV46 interface signals are actively used and which are inactive for the Feature Subset being supported.

2.3 Mirroring /Steering

Feature Subset Applicability: Baseline and Performance

The data bus and BE Mirroring/Steering functionality as defined in the PLB V4.6 specification is required by Xilinx to be incorporated into the PLBV46 interface logic of the attaching peripheral. This logic is not included in the interconnect infrastructure. Parameters have been defined to allow optimization of the logic based on the system design.

2.4 Addressing Restrictions

Feature Subset Applicability: Baseline and Performance

The PLB V4.6 Specification provides for up to 64 bits of addressing that is split between two 32-bit address buses. Xilinx soft IP and EDK tooling will only utilize 32 bits in the initial PLB V4.6 implementation. The PLB V4.6 upper address bus will be included in the required port interface for peripherals but will be driven to zeroes by Masters (M_UABus[0:31]) and internally ignored by Slaves (PLB_UABus[0:31]).

2.5 Transfer Type Restrictions

Feature Subset Applicability: Baseline and Performance

Xilinx soft IP will only support Memory transfer types. Masters will drive M_type(0:2) to "000".

2.6 Parity

Feature Subset Applicability: Baseline and Performance

Xilinx soft IP will not support PLB V4.6 Parity. Neither Masters nor Slave will include the associated PLB V4.6 Parity signaling in their PLB V4.6 interfaces. In addition, the PLB V4.6 Bus/Arbiter will not provide the Parity signals in the Master/Slave attach points and internal routing/logic.

2.7 Transfer Attributes

Feature Subset Applicability: Baseline and Performance

Xilinx soft IP do not support PLB V4.6 transfer attributes. Soft Masters will include the M_Tattribute(0:15) output port in the PLB V4.6 interface but will drive the associated signals to all zeros. Slaves will include the PLB_Tattribute(0:15) input port but will ignore it internally.

- If a Slave design intends to do speculative reads then the designer must be aware that the Guarded Memory attribute is not available and that speculative reads into "non-well behaved" memory may occur. The Slave designer must address this condition as deemed appropriate.
- PPC405 (PLBv46 version) produces values on some TAttribute channels. The PLBv46 bus/arbiter will distribute these values to the Slave ports. However, Slave IP in the EDK library or the slave interfaces of custom IP generated automatically by EDK (Create IP Wizard based on Slave IPIF) will not use of any of these signals.

2.8 Aborts

Feature Subset Applicability: Baseline and Performance

Aborts are not supported. Soft PLBV46 Masters are prohibited from generating aborts. Soft PLBV46 Slave interfaces will ignore any assertions on the PLB_abort input port.

- The PPC405-virtex4 will suppress aborts on the IPLB interface. Aborts on the DPLB interface may occur only as a result of a processor reset, at which time the connected PLB buses and peripherals must also be reset, thus avoiding any need for any PLBv46-based IP to respond to the M_Abort or PLB_Abort signal.

2.9 Cacheline Transfers

Feature Subset Applicability: Performance

Cacheline transfers of 4 words and 8 words are supported. **Cacheline transfers of 16 words are not supported**. Slaves will return Cacheline Read data in sequential address order starting either from the specified target word position or the beginning of the Cacheline.

2.10 Indeterminate Burst Support

Feature Subset Applicability: Performance

Indeterminate Length Bursts are not supported. Masters are prohibited from generating Indeterminate Burst requests. Slaves will ignore Indeterminate Burst requests.

2.11 Fixed Length Burst Support

Feature Subset Applicability: Performance

Fixed length burst protocol will be supported with allowed data beat values of 2 to 16 and allowed transfer width of words (32 bits), double words (64 bits), and quad words (128 bits). Fixed Length Burst requests with data beat counts from 17 to 256 and transfer widths of bytes or half words will be ignored by Xilinx soft IP Slaves. Masters are prohibited from generating Fixed Length Burst requests with data beat counts greater than 16.

2.12 Premature Burst Terminations

Feature Subset Applicability: Performance

Premature Burst terminations are prohibited for both Masters and Slaves. A premature burst termination is defined as the stoppage of a burst transfer prior to the full completion of the data transfer as requested by the Master via the transfer qualifiers asserted during the Address Phase of the request. A full transfer is defined as the requested data beat count multiplied by the requested transfer width (in word units).

2.13 Address Pipelining

Feature Subset Applicability: Performance

Address pipelining support is optional for Slaves. The PLBV46 Bus/Arbiter will support a 2-level address pipelining model. In general, Xilinx soft Slave interfaces do not support address pipelining.

2.14 Pending Request Status

Feature Subset Applicability: Baseline and Performance

Slaves and Masters will ignore the pending request input status but the signals will be present on their PLB V4.6 interfaces: These signals are:

PLB_reqPri(0:1)

PLB_rdPendPri(0:1)

PLB_wrPendPri(0:1)

PLB_rdPendReq

PLB_wrPendReq

2.15 Slave to Master Interrupts

Feature Subset Applicability: Baseline and Performance

Slave to Master Interrupts are not supported.

2.16 Known PLB V4.6 Transfer Protocol Exceptions

The following bullet items identify potential PLBV4.6 protocol violations that result from the Xilinx PLBV4.6 simplifications.

- Bus Arbitration Cycle Timing:
  In a point to point topology, the lack of the need for bus arbitration allows for 0 cycle latency of the PLB Address phase. The PLB BFM Monitor may or may not issue a warning on this.
- Aborts:
  Aborts are not supported. If a system includes a PLB Master that issues aborts to IP following the conventions of this specification, undefined operation will occur and protocol violations may be reported by the PLB BFM Monitor.
- Master Write data bus Mirroring:
  This specification allows for a Master to simplify Write Data bus Mirroring/Steering based on knowledge of the interface data width, the native data width of the Master, and the smallest data width of any Slave the Master may access via the interconnect. This implementation simplification may lead to data mirroring and steering warnings by the PLB BFM Monitor.

2.17 Parameterization

2.17.1 Parameter Naming Conventions

IP Devices connected to the PLB V4.6 need to know the characteristics of the interface attachment. This allows the IP to tailor port sizes that change from implementation to implementation. These are passed into the IP via VHDL Generic parameters. It is possible for an IP to have more than one Slave or Master port attachment. A multi-port memory controller is a good example of an IP the can have many slave ports. In order to parameterize the characteristics of each port individually, a set of Parameters for each port may be required. In this case, the following parameter naming conventions apply.

- The general rule for Interface parameter name becomes C_<busifname>_<parameter_name>. The default values for <busifname> are SPLB and MPLB for a Slave and Master respectively but the tag may be whatever is descriptive for the design and unique for the interface. The only requirement is that once the <busifname> tag is chosen, it is consistently used for all parameters and port names relating to that particular PLBV46 interface.
  - Example: The old parameter name C_PLB_DWIDTH will become (using the default <busifname> tag) C_SPLB_DWIDTH for PLBV46 Slaves and C_MPLB_DWIDTH for PLBV46 Masters.
- If there are multiple PLBV46 Slave Interfaces on the same IP, then a more descriptive <busifname> may be used.
  - Example:A multi-port memory controller designer chooses to use **PORTx** as the <busifname> tag where x is a numerical identifier of each port. The PLBV46 interface data width parameters would be named C_**PORT0**_DWIDTH, C_**PORT1**_DWIDTH, and so on. The same convention holds for multiple master ports and the associated master port parameters.

Some pre-defined general use parameters are detailed in the following section.

2.17.2 Predefined Parameter Names and definitions

The following section lists predefined parameter names that will be recognized by EDK tooling. This is useful from the standpoint that the parameters can be assigned values automatically by EDK tooling during system creation. The values assigned are derived from the characteristics of the system that is specified by the User. **Individual usage requirements of the parameters are delineated in the Master or Slave section of the feature subset definition for each**.

2.17.2.1 C_<busifname>_MID_WIDTH

Applicability: All Slaves

Usage: HDL Required

This integer parameter indicates to a PLBV46 Slave interface the number of bits required for the PLB_masterID input bus for Slave devices. This value varies from one implementation to another based on the number of PLB Masters attached to the connecting PLB. It is an integer value equal to log2(**C**_<busifname>_**NUM_MASTERS**).

2.17.2.2 C_<busifname>_NUM_MASTERS

Applicability: All Slaves

Usage: HDL Required

This integer parameter indicates to a PLBV46 Slave interface the number of PLB Masters present. This value varies from one implementation to another based on the number of PLB Masters attached to the connecting PLB. It is used to set the port widths of the Sl_MIRQ, Sl_Mrderr, Sl_wrerr, and the Sl_busy output ports from Slave interfaces. Defined values are integer 1 to the number of Masters attached to the bus instance connected to the applicable Slave peripheral.

2.0.0.1 C_<busifname>_DWIDTH

Applicability: Slaves and Masters

Usage: HDL Required

This integer parameter is used to indicate the bit width of the data bus attachment to the implemented PLB Bus/Arbiter. The tools set this value based on the maximum native data width of all IP connected to the same bus. Slaves and Masters must incorporate the necessary mirroring and steering interface logic depending on the relationship between their native data width and the bus attachment width designated by this parameter. Defined values are 32, 64 and 128.

2.0.0.2 C_<busifname>_NATIVE_DWIDTH

Applicability: All Slaves and Masters

Usage: HDL Optional, MPD Required

This integer parameter is used to indicate the native bit width of the internal data bus of the peripheral. Some peripherals will require the value of this parameter to be fixed and while other peripherals such as a memory controller port might have selectable native data widths. Defined values are integer 32, 64, and 128.

2.0.0.3 C_<busifname_SUPPORT_BURSTS

Applicability: Performance Slaves

Usage: HDL Optional, MPD Required

This integer parameter is used to allow selection of the supported feature subset for the associated PLBV46 interface on those Slaves that allow it. Some peripheral Slaves will not have feature subset selection while other peripherals such as a memory controller port might have selectable feature subset support on a port by port basis. Defined values are integer 0 and 1 where 0 indicates a Single Data beat only optimization used to conserve resources.

- This is a dual-purpose parameter. Specifying this parameter in the Slave's HDL identifies the Slave to the EDK tools being able to support the Performance Feature subset. This may, in turn, allow some features of other IP connected to the same bus to be appropriately scaled/optimized. If defined, the value of the parameter is always set by EDK tools, depending on the requirements of masters connected to the same bus. Usage of the parameter value by Performance slaves is optional, depending on whether the slave is designed to optimize its resource utilization depending on whether burst support is required by any connected masters.
- **Special Note:** A Performance Slave that does not have an HDL specified parameter C_<busifname>_SUPPORT_BURSTS must include a non-HDL parameter in the MPD file indicating it is a Performance Slave via a constant assignment to a value of 1.

2.0.0.4 C_<busifname>_P2P

Applicability: Slaves and Masters

Usage: HDL Optional

This optional integer parameter is used to indicate to a Slave/Master interface when it is exclusively attached to a PLBV46 Master/Slave via a Point to Point interconnect scheme. In this scenario, the Slave interface may be able to reduce resource utilization by eliminating the address decode function and modifying interface behavior to allow for a reduction in latency. The Master Interface may be able to optimize latency and other behaviors based on point to point topology. Defined values are integer 0 and 1 where 0 indicates the shared bus topology and 1 indicates a point to point topology.

2.17.2.2.1 C_<busifname>_SMALLEST_MASTER

Applicability: Performance Slaves

Usage: HDL Optional

This integer parameter is used by 64-bit and 128-bit Slave devices and it indicates the smallest native bit-width of any Master that may access it. This parameter allows optimizations within the Slave interface logic (i.e byte enable muxing and data steering) if narrower masters don’t have to be supported for that application. Defined values are integer 32, 64, and 128.

2.17.2.3 C_<busifname>_SMALLEST_SLAVE

Applicability: Performance Masters

Usage: HDL Optional

This integer parameter is used by 64-bit and 128-bit Master devices and it indicates smallest native bit-width of any Slave on the corresponding PLB V46 Bus with which the Master will interact. This parameter allows the means for optimizations within the Master interface logic (i.e. Data mirroring granularity) if Slaves of narrower width don’t have to be supported for that application. Defined values are integer 32, 64, and 128.

2.17.2.4 C_<busifname>_CLK_PERIOD_PS

Applicability: Slaves and Masters

Usage: HDL Optional

This is an optional parameter that is used to specify the period of the PLBV46 bus clock (in picoseconds) for the corresponding PLBV46 interface attachment. It has been defined for use by IP that needs to know the Bus clock rate to implement certain functions such as internal timers.

2.17.3 PLBV46 Port Naming

As stated earlier in this document, all PLBV46 interfaces incorporate the same signal set regardless of feature subset supported. The advent of a single IP design that has multiple PLBV46 interfaces requires unique delineation of the interfaces for support of tool automation of the system interconnect.

2.17.3.1 PLBV46 Interface Port Naming Convention

The general rule for Interface port name becomes <**busifname**>_<**plbv46signame**>. The <busifname> tag is optional for peripherals that only have a single Slave interface and/or a single Master interface. If the <busifname> tag is omitted, the interconnect tools assume a default value for <busifname> of SPLB and MPLB for a Slave and Master respectively.

- Slave Example: A multi-port memory controller designer decides to use explicit <busifname> of PORTx where x is a numerical delineation for each of the PLBV46 Slave interface ports. Then all PLBV46 interface port names would be of the form PORTx_<plbv46signame>. i.e. **PORT0**_PLB_PAValid**, PORT1**_PLB_PAValid, **PORT0**_Sl_addrAck, **PORT1**_Sl_addrAck, **PORT0**_Clk, **PORT1**_Clk, **PORT0**_Rst**, PORT1**_Rst.
- Master Example: A microprocessor design has two PLBV46 Master Interfaces, one for Instruction fetching and one for Data accesses, The <busifname> tag is chosen to be **IPLB** for the instruction side and **DPLB** for the Data side. Some example port names

would be: **IPLB**_M_request, **DPLB**_M_request, **IPLB**_PLB_MaddrAck, **DPLB**_PLB_MBusy.

Section 3

Baseline Peripheral Requirements

3.1 Baseline Peripheral Use Case

The Baseline feature subset is targeted to PLBV46 Bus interface applications that only require single data beat accesses that transfer up to 4 bytes (32-bits) at a time. This is generally the case for control and status register accesses and low data rate transfer for peripherals like a UART or GPIO. The focus of the Baseline feature definition is simplicity and resource efficiency.

3.2 Baseline Slave Feature Subset Summary

The following two sections delineate the supported and unsupported PLBV46 features applicable to the Baseline Slave simplifications..

3.2.1 Supported Functions

- 32-bit addressing
- 32-bit native data width
    - Sl_Ssize = "00" 32-bits only
- PLB single data beat transfer requests of 1-4 bytes
- Address Phase Rearbitrate
- Address Phase Wait
- Data Phase Busy Indication
- Only supports Memory transfer type (PLB_type(0:2) = "000")
- Connects to 32, 64, or 128-bit PLBV46 interface
    - If connected to 64 or 128 bit PLBV46, then the Baseline Slave is required to include the input PLB_BE muxing and Sl_rdDBus data bus mirroring as per IBM CoreConnect requirements. The need for this logic is inferred from the value of the the C_<busifname>_DWIDTH parameter which indicates the bit width of the PLBV46 interface.
- Read and Write Error Generation is allowed if required by the IP application
- Inhibit Sl_addrack assertion when receiving an unsupported transfer request

- i.e. Any burst or cacheline request or non-memory type qualifier

3.2.2 Unsupported Functions

The following functions and behaviors are prohibited:

- Aborts
- Address pipelining
- Slave support for Split bus
- Non-Memory transfer types (DMA Flyby, DMA Buffered External, etc)
- Indeterminate Burst requests
- Fixed Length Burst Requests
- Cacheline Requests
- Parity
- Transfer Attributes
- Pending Request and Priority input information
- Slave to Master Interrupts
- Data steering (not required for 32-bit slaves)

3.3 Baseline Slave PLBV46 Interface Signal Set and Usage

The following table defines the PLBV46 Baseline Slave signal set usage. **All signals listed are required to be present on the PLBV46 interface**. Note that the <busifname> prefix is not shown for clarity except in the case of the SPLB_Clk and SPLB_Rst ports. The Usage Notes column indicates how the Baseline Slave is to use/drive the associated signal.

Table 3-1: **Baseline Slave Implementation Signal Usage**

Signal Name	IF Type	iO Direction	Usage Notes
SPLB_Clk	Slave	I	Used
SPLB_Rst	Slave	I	Used
PLB_abort	Slave	I	Port required but ignore internally
PLB_ABus(0:31)	Slave	I	Used
PLB_UABus(0:31)	Slave	I	Port required but ignore internally
PLB_BE(0:**C_<busifname>_DWIDTH**/8-1)	Slave	I	Used
PLB_busLock	Slave	I	Port required but ignore internally
PLB_lockErr	Slave	I	Port required but ignore internally
PLB_masterID(0:**C_<busifname>_MID_WIDTH**-1)	Slave	I	Used

Table 3-1: **Baseline Slave Implementation Signal Usage**

Signal Name	IF Type	iO Direction	Usage Notes
PLB_PAValid	Slave	I	Used
PLB_rdPendPri(0:1)	Slave	I	Port required but ignore internally
PLB_wrPendPri(0:1)	Slave	I	Port required but ignore internally
PLB_rdPendReq	Slave	I	Port required but ignore internally
PLB_wrPendReq	Slave	I	Port required but ignore internally
PLB_rdBurst	Slave	I	Port required but ignore internally
PLB_rdPrim	Slave	I	Port required but ignore internally
PLB_reqPri(0:1)	Slave	I	Port required but ignore internally
PLB_RNW	Slave	I	Used
PLB_SAValid	Slave	I	Port required but ignore internally
PLB_Msize(0:1)	Slave	I	Port required but ignore internally
PLB_size(0:3)	Slave	I	Used (Slave only issues Sl_addrAck when the value is "0000" and there is an address decode hit)
PLB_TAttribute	Slave	I	Port required but ignore internally
PLB_type(0:2)	Slave	I	Port required but ignore internally
PLB_wrBurst	Slave	I	Port required but ignore internally
PLB_wrDBus(0:**C_<busifname>_DWIDTH**-1)	Slave	I	Used
PLB_wrPrim	Slave	I	Port required but ignore internally
Sl_addrAck	Slave	O	Used (see PLB_size(0:3) Usage note)
Sl_MBusy(0:**C_<busifname>_NUM_MASTERS**-1)	Slave	O	Used
Sl_MRdErr(0:**C_<busifname>_NUM_MASTERS**-1)	Slave	O	Used if Slave has error condition otherwise drive to '0'

Table 3-1: **Baseline Slave Implementation Signal Usage**

Signal Name	IF Type	iO Direction	Usage Notes
Sl_MWrErr(0:**C_<busifname>_NUM_MASTERS**-1)	Slave	O	Used if Slave has error condition otherwise drive to '0'
Sl_MIRQ(0:**C_<busifname>_NUM_MASTERS**-1)	Slave	O	Drive to logic low)
Sl_rdBTerm	Slave	O	Drive to logic low)
Sl_rdComp	Slave	O	Used (For simplicity, drive with Sl_rdAck)
Sl_rdDAck	Slave	O	Used
Sl_rdDBus(0:**C_<busifname>_DWIDTH**-1)	Slave	O	Used
Sl_rdWdAddr(0:3)	Slave	O	Drive to logic low)
Sl_rearbitrate	Slave	O	Used
Sl_SSize(0:1)	Slave	O	Used but fixed value of "00" (always 32-bit native data width)
Sl_wait	Slave	O	Used
Sl_wrBTerm	Slave	O	Drive to logic low)
Sl_wrComp	Slave	O	Used (For simplicity, drive with Sl_wrAck)
Sl_wrDack	Slave	O	Used

3.4 Baseline Slave Parameterization

The following parameters are required for each PLBV46 Baseline Slave interface.

- Required by EDK tooling
  - C_<busifname>_NATIVE_DWIDTH (Optional in HDL, Required in MPD, fixed at value of 32)
- Required to configure PLBV46 Interface
  - C_<busifname>_DWIDTH
  - C_<busifname>_NUM_MASTERS
  - C_<busifname>_MID_WIDTH
- Depending on the requirements of the Slave device, the following predefined parameters are optionally available for use:
- C_<busifname>_CLK_PERIOD_PS
- C_<busifname>_P2P

3.5 Baseline Slave Interconnect Considerations

3.5.1 Point to Point Interconnect Optimizations

The Baseline Slave may optionally optimize PLBV46 interface behavior such address decoding removal and latency reduction (due to interface registering) connected directly to a Master via a point to point interconnect scheme. Point to Point topology will be indicated to the Slave interface via assignment of a value of 1 to the parameter C_<busifname>_P2P. The assertion of Sl_rearbitrate by the Slave in a point to point topology is not meaningful because there is no arbitration logic in the interconnect. Assertion of Sl_wait is more optimal in this topology.

3.5.1.1 Point to Point Interconnect Considerations

A PLBV46 Master and a PLBV46 Slave that are connected via a Point to Point scheme will have the same Native Data Width. This avoids the need for the Master to support Conversion Cycles and for both the Master and Slave eliminates the need to support Burst length expansion.

3.5.2 Shared Bus Interconnect

Shared Bus interconnect generally requires additional registering of inputs and outputs on the PLBV46 interface to avoid Fmax issues caused by excessive fanout and loading of shared interconnect signals. A general rule of thumb is to register all inputs and outputs from/to the PLBV46 interface however some signals cannot be registered due to required PLB transfer protocol.

3.6 Baseline Slave Normal Transfers

A normal transfer for a Baseline Slave is defined as a memory type transfer of a single data beat read or write request of 1 to 4 bytes.

3.7 Baseline Slave Exceptions

Baseline Slaves are required to inhibit assertion of the SL_addrAck when an access is attempted with an unsupported request. This will cause the PLBV46 Bus/Arbiter to Timeout the transfer attempt and assert a logic high on the detected error output pin. This provides means by which Users may track error sources within a system should a Master attempt illegal operation with a Baseline Slave.

3.8 Baseline Master Feature Subset Summary

3.8.1 Supported Functions

- 32-bit addressing
- 32-bit native data width
    - ♦ M_MSize = “00” (32-bits only)
- PLB Single data beat transfer requests of 1 to 4 bytes
- Only supports Memory transfer type (M_type(0:2) = “000”)

- Connects to 32, 64, or 128-bit PLBV46 interface
    - Mirroring connection scheme must be included in the Master's write data bus interface when attaching to a PLBV46 wider than 32-bits
    - Read and Write Error reply from a Slave device may be optionally used if needed by the IP Core

3.8.2 Unsupported Functions

The following functions and behaviors are prohibited:

- Generation of Aborts
- Address pipelining and Split bus
- Generation of Non-Memory transfer types (DMA Flyby, DMA Buffered External, etc)
- Generation of Indeterminate Burst requests
- Generation of Fixed Length Burst Requests
- Generation of Cacheline Requests
- Use of Parity
- Use of Transfer Attributes
- Use of Pending Request and Priority input information
- Use of Slave to Master Interrupts

3.9 Baseline Master PLBV46 Interface Signal Set and Usage

3.9.1 PLBV46 Interface Signal Set

The following table defined the PLBV46 Baseline Master signal set usage. **All signals listed are required to be present on the PLBV46 Master interface**. Note that the <busifname> prefix is not shown for clarity except for the MPLB_Clk and MPLB_Rst input signals. The Usage Notes column indicates how the Baseline Master is to use/drive the associated signal.

Table 3-2: **Baseline Master Implementation Signal Set and Usage**

Signal Name	IF Type	IO Direction	Usage Notes
MPLB_Clk	Master	I	Used
MPLB_Rst	Master	I	Used
PLB_MBusy	Master	I	Used
PLB_MRdErr	Master	I	Used if needed otherwise ignore internally
PLB_MWrErr	Master	I	Used if needed otherwise ignore internally
PLB_MIRQ	Master	I	Port required but ignore internally

Table 3-2: **Baseline Master Implementation Signal Set and Usage** ***(Contd)***

Signal Name	IF Type	IO Direction	Usage Notes
PLB_MWrBTerm	Master	I	Port required but ignore internally
PLB_MWrDAck	Master	I	Used
PLB_MAddrAck	Master	I	Used
PLB_MRdBTerm	Master	I	Port required but ignore internally
PLB_MRdDAck	Master	I	Used
PLB_MRdDBus(0:**C_<busifname>_ DWIDTH**-1)	Master	I	Used
PLB_MRdWdAddr(0:3)	Master	I	Port required but ignore internally)
PLB_MRearbitrate	Master	I	Used if Master will be implementing Bus lock operations, otherwise ignore internally
PLB_MSSize(0:1)	Master	I	Port required but ignore internally
PLB_MTimeout	Master	I	Used
M_abort	Master	O	Drive to logic low)
M_ABus(0:31)	Master	O	Used
M_UABus(0:31)	Master	O	Drive to logic low)
M_BE(0:**C_<busifname>_ DWIDTH**/8-1)	Master	O	Used
M_busLock	Master	O	Drive to logic low)
M_lockErr	Master	O	Drive to logic low)
M_MSize(0:1)	Master	O	Used Drive to “00” for 32-bit Master
M_priority(0:1)	Master	O	Drive to logic low)
M_rdBurst	Master	O	Drive to logic low)
M_request	Master	O	Used
M_RNW	Master	O	Used
M_size(0:3)	Master	O	Used
M_TAttribute(0:15)	Master	O	Drive to logic low)
M_type(0:2)	Master	O	Drive to logic low)

Table 3-2: **Baseline Master Implementation Signal Set and Usage *(Contd)***

Signal Name	IF Type	IO Direction	Usage Notes
M_wrBurst	Master	O	Drive to logic low)
M_wrDBus(0:**C_<busifname>_DWIDTH**-1)	Master	O	Used

3.9.2 Additional Baseline Master Signals

The PLBV46 Baseline Master is also required to include a detected error output port that is named MD_error. This output is not part of the PLBV46 interface but is deemed necessary for system troubleshooting support. This output is asserted active high and then held until the Master is reset whenever the Master encounters an error condition that is deemed important to the Master application. Examples of such error conditions are detected assertions of the PLB_MRdErr, PLB_MWrErr, or PLB_MTimeout. Internal error conditions may also be used to assert the MD_error output. If the Master design chooses not to support error detection, then this signal will be driven to logic low.

3.10 Baseline Master Parameterization

The following parameters are required for each PLBV46 Baseline Master interface.

- Required by EDK tooling
  - C_<busifname>_NATIVE_DWIDTH (Optional in HDL, Required in MPD, fixed at value of 32)
  - MPD file must contain a non-HDL parameter **GENERATE_BURSTS** that is set to **FALSE**. See Section 6, "EDK Engines (platgen) Support for Parameters"
- Required to configure PLBV46 Interface
  - C_<busifname>_DWIDTH

Depending on the requirements of the Master device, the following predefined parameters are available for use:

- C_<busifname>_CLK_PERIOD_PS
- C_<busifname>_P2P

3.11 Baseline Master Interconnect Requirements

3.11.1 Point to Point Interconnect

The Baseline Master may optionally optimize PLBV46 interface behavior such latency reduction (due to interface registering) connected directly to a Slave via a point to point interconnect scheme. Point to Point topology will be indicated to the Master interface via the value assigned to the parameter C_<busifname>_P2P.

3.11.1.0.1 Point to Point Interconnect Considerations

A PLBV46 Master and a PLBV46 Slave that are connected via a Point to Point scheme will have the same Native Data Width. This avoids the need for the Master to support

Conversion Cycles and for both the Master and Slave eliminates the need to support Burst length expansion.

3.11.2 Shared Bus Interconnect

Shared Bus interconnect generally requires additional registering of inputs and outputs on the PLBV46 interface to avoid Fmax issues caused by excessive fan out and loading of shared interconnect signals. A general rule of thumb is to register all inputs and outputs from/to the PLBV46 interface however some signals cannot be registered due to required PLB transfer protocol.

3.12 Baseline Master Normal Transfers

A normal transfer for a Baseline Master is defined as a memory type transfer of a single data beat read or write request of 1 to 4 bytes.

3.13 Baseline Master Exceptions

Should a Baseline Master encounter an exception, such as a PLB_Timeout assertion or assertion of the PLB_Mrderr or PLB_Mwrerr by the target Slave, the Master is required to assert and hold the Detected Error output signal (MD_error). This assertion is maintained until the Master is reset.

Performance Peripheral Requirements

4.1 Use Case

The Performance feature subset is targeted to PLBV46 Bus interface applications that require high performance data transfers (like DMA devices) or are Processors that require caching support.

4.2 Performance Slave Feature Subset

4.2.1 Supported Functions

- 32-bit addressing
- 32, 64, or 128-bit native data widths allowed
- Attachment to a 32, 64, or 128-bit PLBV46 Interface
  - PLBV46 interface data width is set via the C_<busifname>_DWIDTH parameter value
  - Sl_Ssize driven to PLBV46 reflects native data width of the Slave design irrespective of the PLBV46 Interface attachment width
  - The Slave cannot be attached to a PLBV46 Interface that is narrower than the native data width of the Slave
- Address Phase Rearbitrate
- Address Phase Wait
- Memory transfer type only (PLB_type(0:2) = "000")
- Singles data beat transfer of 1 to 4 bytes (32-bit native data width), 1 to 8 bytes (64-bit native data width, or 1 to 16 bytes (128-bit native data width)
- Fixed Length Burst Transfer (2 to 16 data beats only)
  - Supported transfer size of words, double words, and quad words
  - The Performance Slave is required to assert the Sl_rdbterm and the Sl_wrbterm on the next to last data beat of the requested fixed length burst transfer
  - The Performance Slave is not allowed to do early burst termination of Fixed Length Bursts

    - **Wide Master and Narrow Slave design consideration:** A situation arises when a wide Master generates Fixed Length Burst request of a data size (as indicated by PLB_size) that exceeds the native width of 32-bit and 64-bit Performance Slaves. In this case, the number of data beats that are required to complete the requested transfer is greater (2x or 4x) than the explicitly requested data count (2 to 16) as indicated by the Byte Enable signals. To avoid premature burst termination, Performance Slaves **are required to dynamically adjust their burst run length** to account for this condition. In this case it is possible to have an explicit data beat count for the request be 16 or less data beats but the requested data width of the transfer by the Master (for example an 8 Quadword Fixed Length Burst request to a 32-bit Performance Slave) may require more than 16 data beats. In this example, there are 4 words per Quadword * 8 data beats = 32 words being requested which requires 32 data beats from a 32-bit Performance Slave.
        - **32-bit Performance Slave design consideration**: Performance Slaves that are natively 32-bits, are only required to sample BE 0:3 during Fixed length burst requests. Since the explicit Fixed length Burst transfer count is limited to 16 for any Master designed using the Xilinx simplifications, this does not cause any ambiguity.
    - It is allowed for a Master to request a Fixed Length Burst transfer that has a transfer size (PLB_size(0:3)) that is narrower than the indicated size of the Master on PLB_Msize (0:1) port. Fixed Length Burst requests with transfer sizes wider than the Master's indicated data width (PLB_Msize(0:1)) are not allowed. Xilinx Simplifications require 64-bit and 128-bit Masters to drive their respective M_BE(4:7) bits with zeros during Fixed Length Bursts (M_BE(0:3) indicate the data beat count of 2 to 16).
        - A Performance Slave must not respond if PLB_BE(4:7) are non-zero during a Fixed Length Burst request and the indicated Master size is 64 or 128 bits.
        - A Performance Slave attached to a 64-bit or 128-bit PLB will ignore PLB_BE(4:7) during a Fixed Length Burst request that is from a 32-bit Master (PLB_Msize(0:1)='00").
        - A Performance Slave is required to adhere to the Read Data Bus Steering requirements that are based on the connecting PLB Bus data width and the data beat address (transfer size independent).
- Cacheline transfer of 4 or 8 words
    - 16 Word Cacheline transfers are not supported
    - The Slave will drive the Sl_rdWdAddr bus in accordance with Target word first or Line-aligned word first depending on the Cacheline Read data reply ordering being used by the Slave. For either method, the remaining words of the Cacheline must be transferred in sequential order (with wrap-around as needed).
    - **Mixed Width Master and Slave design consideration:** Slaves are required to adjust the number of required data beats needed (based on the native data width of the Slave and the reported Master data width (PLB_MSize(0:1)) to accommodate the requested Cacheline transfer.
- Read and Write Error Generation allowed if required by the IP Core
    - If the Slave does not have error conditions, then the SL_rderr and the SL_wrerr should be driven to logic low
    - Read Bus Data steering is required when a Performance Slave is accessed by a narrower Master (this applies to native 64-bit and 128-bit Performance Slave designs only)

    - Read data bus steering may be optimized using the optional parameter C_SMALLEST_MASTER. If the Slave will not be accessed by a smaller data width Master in the system design, then the steering logic provides no purpose and can be eliminated or minimized.

4.2.2 Unsupported Functions

The following functions and behaviors are prohibited:

- Aborts
- Non-Memory transfer types (DMA Flyby, gDMA Buffered External, etc. are ignored)
- Fixed Length Burst transfer requests of 17 to 256 data beats
- Fixed Length Bursts of size byte and half word
- Premature Fixed Length Burst terminations
- Indeterminate Burst Transfers
- Cacheline transfers of 16 words
- Parity
- Transfer Attributes
- Pending Request and Priority input information
- Slave to Master Interrupts

4.2.3 Optional Functions

The following features are allowed but not normally supported by most Performance Slaves due to complexity and resource utilization requirements:

- Address Pipelining (2-level only)
    - Note: This requires the use of the PLB_SAValid, PLB_rdprim, and PLB_wrprim signals by the Slave
- Split bus Architecture (simultaneous read/write support)

4.2.4 Performance Slave Signal Set and Usage

The following table defines the PLBV46 Performance Slave signal set usage. **All signals listed are required to be present on the PLBV46 Slave interface.** Note that the <busifname> prefix is not shown for clarity except for the SPLB_Clk and the SPLB_Rst signals. The Usage Notes column indicates how the Performance Slave is to use/drive the associated signal.

Table 4-1: **Slave Signal Usage**

Signal Name	IF Type	IO Direction	Usage Notes
SPLB_Clk	Slave	I	Used
SPLB_Rst	Slave	I	Used
PLB_abort	Slave	I	Port required but ignore internally
PLB_ABus(0:31)	Slave	I	Used
PLB_UABus(0:31)	Slave	I	Port required but ignore internally

Table 4-1: **Slave Signal Usage *(Contd)***

Signal Name	IF Type	IO Direction	Usage Notes
PLB_BE(0:**C_<busifname>_DWIDTH**/8-1)	Slave	I	Used
PLB_busLock	Slave	I	Used if needed
PLB_lockErr	Slave	I	Used if needed
PLB_masterID(0:**C_<busifname>_MID_WIDTH**-1)	Slave	I	Used
PLB_PAValid	Slave	I	Used
PLB_rdPendPri(0:1)	Slave	I	Port required but ignore internally
PLB_wrPendPri(0:1)	Slave	I	Port required but ignore internally
PLB_rdPendReq	Slave	I	Port required but ignore internally
PLB_wrPendReq	Slave	I	Port required but ignore internally
PLB_rdBurst	Slave	I	Used
PLB_rdPrim	Slave	I	Port required. **If optional Address Pipelining is implemented, then this signal is used**. Otherwise, ignore internally
PLB_reqPri(0:1)	Slave	I	Port required but ignore internally
PLB_RNW	Slave	I	Used
PLB_SAValid	Slave	I	Port required. **If optional Address Pipelining is implemented, then this signal is used**. Otherwise, ignore internally
PLB_Msize(0:1)	Slave	I	Used (if narrower Master accesses Slave, Slave must steer read data bus appropriately)
PLB_size(0:3)	Slave	I	Used but allowed values are "0000" (for singles), “0001” (4 wrd cacheline, “0010” (8 wrd cacheline), "1010", "1011", or "1100" (fixed length bursts of data width less than or equal to the requesting Master’s native data width)
PLB_TAttribute	Slave	I	Port required but ignore internally
PLB_type(0:2)	Slave	I	Port required but ignore internally
PLB_wrBurst	Slave	I	Used
PLB_wrDBus(0:**C_<busifname>_DWIDTH**-1)	Slave	I	Used

Table 4-1: **Slave Signal Usage** ***(Contd)***

Signal Name	IF Type	IO Direction	Usage Notes
PLB_wrPrim	Slave	I	Port required. **If optional Address Pipelining is implemented, then this signal is used**. Otherwise, ignore internally
Sl_addrAck	Slave	O	Used
Sl_MBusy(0:**C_<busifname>_NUM_MASTERS**-1)	Slave	O	Used
Sl_MRdErr(0:**C_<busifname>_NUM_MASTERS**-1)	Slave	O	Used if Slave has read error condition requirements otherwise drive to '0'
Sl_MWrErr(0:**C_<busifname>_NUM_MASTERS**-1)	Slave	O	Used if Slave has write error condition requirements otherwise drive to '0'
Sl_MIRQ(0:**C_<busifname>_NUM_MASTERS**-1)	Slave	O	Drive to logic low
Sl_rdBTerm	Slave	O	Used for normal Fixed Length Burst read termination indication. **Premature burst termination is prohibited**.
Sl_rdComp	Slave	O	Used
Sl_rdDAck	Slave	O	Used
Sl_rdDBus(0:**C_<busifname>_DWIDTH**-1)	Slave	O	Used
Sl_rdWdAddr(0:3)	Slave	O	Used for Cacheline Read support
Sl_rearbitrate	Slave	O	Used
Sl_SSize(0:1)	Slave	O	Used
Sl_wait	Slave	O	Used
Sl_wrBTerm	Slave	O	Used for normal Fixed Length Burst write termination indication. **Premature burst termination is prohibited**.
Sl_wrComp	Slave	O	Used
Sl_wrDack	Slave	O	Used

4.2.5 Performance Slave Parameterization

The following parameters are required for each PLBV46 Performance Slave interface.

- Required by EDK tooling
  - C_<busifname>_NATIVE_DWIDTH (Optional in HDL, Required in MPD)
  - C_<busifname>_SUPPORT_BURSTS (Optional in HDL, Required in MPD)

- Required to configure PLBV46 Interface
    - C_<busifname>_DWIDTH
    - C_<busifname>_NUM_MASTERS
    - C_<busifname>_MID_WIDTH
    -

Depending on the optimization requirements of the Slave device, the following predefined optional parameters are available for use to internally optimize the Slave interface implementation based on the application :

- C_<busifname>_SMALLEST_MASTER
- C_<busifname>_CLK_PERIOD_PS
- C_<busifname>_P2P

4.2.6 Performance Slave Interconnect Requirements

4.2.6.1 Point to Point Interconnect Optimizations

The Performance Slave may optionally optimize PLBV46 interface behavior such address decoding removal and latency reduction (due to interface registering) connected directly to a Master via a point to point interconnect scheme). Point to Point topology will be indicated to the Slave interface via the value assigned to the parameter C_<busifname>_P2P. The assertion of Sl_rearbitrate by the Slave in a point to point topology is not meaningful since there is no arbitration logic in the interconnect. Assertion of Sl_wait is more optimal in this topology.

4.2.6.1.1 Point to Point Interconnect Considerations

A PLBV46 Master and a PLBV46 Slave that are connected via a Point to Point scheme will have the same Native Data Width. This avoids the need for the Master to support Conversion Cycles and for both the Master and Slave eliminates the need to support Burst length expansion.

4.2.6.2 Shared Bus Interconnect

Shared Bus interconnect generally requires additional register of inputs and outputs on the PLBV46 interface to avoid Fmax issues caused by excessive fanout and loading of shared interconnect signals. A general rule of thumb is to register all inputs and outputs from/to the PLBV46 interface however some signals cannot be registered due to required PLB transfer protocol.

4.2.7 Performance Slave Normal Transfers

The following transfers are considered normal operations for the Performance Slave:

- Single data beat read/write (1 to 16 bytes depending on the Slave's native width)
- Fixed length Burst read/write request of 2 to 16 data beats
    - The actual number of data beats needed to complete the request could be 2x or 4x. This is based on the ratio of the requested transfer size of the burst by the Master relative to the Slave's native data width.
- Cacheline read/write of 4 or 8 words

- The number of actual data beats required is dependent on the smaller of either the Master's or Slave's native data width.

4.2.8 Performance Slave Exceptions

Performance Slaves are required to inhibit assertion of the SL_addrAck when an access is attempted with an unsupported request. This will cause the PLBV46 Bus to Timeout the transfer attempt and assert a logic high on the detected error output pin. This provides means by which Users may track error sources within a system should a Master attempt illegal operation with a Performance Slave.

4.3 Performance Master Feature Subset

4.3.1 Supported Features

- 32-bit addressing
- 32, 64, or 128-bit native data width
    - M_size(0:1) driven out to PLB reflects native data width
- Memory transfer type only (PLB_type(0:2) = "000")
- Singles data beat transfer of 1 to 4 bytes (32-bit native data width), 1 to 8 bytes (64-bit native data width, or 1 to 16 bytes (128-bit native data width)
    - **Wide Masters to Narrow Slave Design Considerations**: Performance Masters that have 64-bit or 128-bit native data widths **must support conversion cycles if** the requested byte enable qualifiers asserted by the Master during a single data beat transfer can cross the native data width alignment of any target Slave the Master may access.
    - If a Master does not support conversion cycles and the requested byte enable qualifiers asserted by the Master during a single data beat transfer do cross the native data width alignment of a narrower Slave responding to the request, then the Master must detect the condition and assert its MD_error output.
- Fixed Length Burst Transfer (2 to 16 data beats)
    - Supported transfer size of words, double words, and quad words but not exceeding the native data width of the Master.
    - **Wide Masters to Narrow Slave Design Considerations**: Performance Masters that have 64-bit or 128-bit native data widths **are required to support dynamic burst length adjustment if**:
        - The requested transfer size for any Fixed Length Burst request exceeds the native data width of the target Slave
    - It is allowed for a Performance Master to request a Fixed Length Burst transfer that has a specified transfer size (M_size(0:3)) that is narrower than the indicated size of the Master on M_Msize (0:1) port.
        - Fixed Length Burst requests with transfer sizes wider than the Master's data width driven on the M_Msize(0:1) output port are not allowed.
        - It is required that 64-bit and 128-bit (Native Data Width) Masters must drive their respective M_BE(4:7) bits with zeros during Fixed Length Bursts.
        - M_BE(0:3) will be driven to indicate the needed data beat count of 2 to 16.
        - A Performance Master is required to adhere to the Write Data Bus Steering (and associated M_BE) requirements that are based on the connecting PLB data width and the data beat address (transfer size independent).
- Write data bus mirroring and steering
- Dynamic request Priority
- Request Timeout
- Request Rearbitration
- Bus Locking

4.3.2 Optional Features

Cacheline read and write requests 4 or 8 words

- These transfers are generally only needed by **processing elements** such as Microblaze.
- The data unit width of each beat of a cache-line transfer will be the smaller of the Master's native width or the responding Slave's native width (as signaled by PLB_MSsize).
- **Wide Masters to Narrow Slave Design Considerations:** Masters are required to adjust (increase) the number of required data beats needed to complete the cacheline transfer when the reported Slave data width (via PLB_MSsize(0:1)) is less than the Master's native data width.
- Generating Cacheline transfer requests require the Master to use the PLB_MRdWdAddr(0:3) input signals and support data transfers ordered sequentially from either the beginning of the cache-line or the unit of data containing the target word.
    - Split bus Architecture (simultaneous read/write support)

4.3.3 Unsupported Features

The following functions and behaviors are prohibited:

- Aborts
- Fixed Length Bursts of length 17 to 256 data beats
- Fixed Length Bursts of size byte and half word
- Indeterminate Length Bursts
- Premature Master Fixed Length Burst terminations
- Performance Masters are prohibited from initiating burst or cacheline transfer requests to Baseline slaves
- Cacheline requests of 16 words
    - A request timeout will occur if this is attempted since the Baseline Slave is required to inhibit Sl_addrAck assertion under these conditions

4.3.4 Performance Master Signal Set and Usage

The following table defines the PLBV46 Performance Master signal set usage. All signals listed are required to be present on the PLBV46 Master interface. Note that the <busifname> prefix is not shown for clarity. The Usage Notes column indicates how the Performance Master is to use/drive the associated signal.

Table 4-2: **Master Interface Signal Usage**

Signal Name	IF Type	IO Direction	Usage Notes
MPLB_Clk	Master	I	Used
MPLB_Rst	Master	I	Used
PLB_MBusy	Master	I	Used
PLB_MRdErr	Master	I	Used if needed otherwise ignore internally

Table 4-2: **Master Interface Signal Usage** ***(Contd)***

Signal Name	IF Type	IO Direction	Usage Notes
PLB_MWrErr	Master	I	Used if needed otherwise ignore internally
PLB_MIRQ	Master	I	Port required but ignore internally
PLB_MWrBTerm	Master	I	Used
PLB_MWrDAck	Master	I	Used
PLB_MAddrAck	Master	I	Used
PLB_MRdBTerm	Master	I	Used
PLB_MRdDAck	Master	I	Used
PLB_MRdDBus(0:**C_<busifname>_DWIDTH**-1)	Master	I	Used
PLB_MRdWdAddr(0:3)	Master	I	Port required. If Cacheline requests are generated by the Master then this input bus is used, otherwise ignore internally
PLB_MRearbitrate	Master	I	Used if Master will be implementing Bus lock operations, otherwise ignore internally
PLB_MSSize(0:1)	Master	I	Used
PLB_MTimeout	Master	I	Used
M_abort	Master	O	Drive to logic low
M_ABus(0:31)	Master	O	Used
M_UABus(0:31)	Master	O	Drive to logic low
M_BE(0:**C_<busifname>_DWIDTH**/8-1)	Master	O	Used
M_busLock	Master	O	Used if needed
M_lockErr	Master	O	Used if needed
M_MSize(0:1)	Master	O	Used Drive to: “00” for 32-bit Master, “01” for 64-bit Master, “10” for 128-bit Master
M_priority(0:1)	Master	O	Used
M_rdBurst	Master	O	Used. Premature burst termination is prohibited

Table 4-2: **Master Interface Signal Usage** ***(Contd)***

Signal Name	IF Type	IO Direction	Usage Notes
M_request	Master	O	Used
M_RNW	Master	O	Used
M_size(0:3)	Master	O	Used but allowed values are "0000" (for singles), “0001” (4 wrd cacheline, “0010” (8 wrd cacheline), "1010", "1011", or "1100" (fixed length bursts of data width that do not exceed either the values of **C_<busifname>_NATIVE_DWIDTH or C_<busifname>_DWIDTH**)
M_TAttribute(0:15)	Master	O	Unused (drive to logic low)
M_type(0:2)	Master	O	Unused (always memory type so drive to logic low)
M_wrBurst	Master	O	Used. Premature burst termination is prohibited
M_wrDBus(0:**C_<busifname>_DWIDTH**-1)	Master	O	Used

4.3.5 Additional Performance Master Signals

The PLBV46 Performance Master is also required to include a detected error output port that is named MD_error. This output is not part of the PLBV46 interface but is deemed necessary for system troubleshooting support. This output is asserted active high and then held until the Master is reset whenever the Master encounters an error condition that is deemed important to the Master application. Examples of such error conditions are detected assertions of the PLB_MRdErr, PLB_MWrErr, or PLB_MTimeout. Internal error conditions may also be used to assert the MD_error output. An example of this is the case where the Master does not support conversion cycles and the requested byte enable qualifiers asserted by the Master during a single data beat transfer cross the native data width alignment of a narrower Slave responding to the request. If the Master design chooses not to support error detection, then this signal will be driven to logic low.

4.3.6 Performance Master Parameterization

The following parameters are required for each PLBV46 Performance Master interface. Note that the ‘x’ in the parameter name is not used if only one Master port is on the IP core.

- Required by EDK tooling
  - C_<busifname>_NATIVE_DWIDTH (Optional in HDL, Required in MPD)
  - MPD file must contain a non-HDL parameter **GENERATE_BURSTS** that is set to **TRUE** . See Section 6, “EDK Engines (platgen) Support for Parameters”.
  - Required to configure PLBV46 Interface
  - C_<busifname>_DWIDTH

Depending on the requirements of the Master device, the following predefined parameters are available for use:

- C_<busifname>_CLK_PERIOD_PS
- C_<busifname>_P2P

4.3.7 Performance Master Interconnect Requirements

4.3.7.1 Point to Point Interconnect Optimizations

The Performance Master may optionally optimize PLBV46 interface behavior such as latency reduction (due to interface registering) connected directly to a Slave via a point to point interconnect scheme). Point to Point topology will be indicated to the Master interface via the value assigned to the parameter C_<busifname>_P2P.

4.3.7.1.1 Point to Point Interconnect Considerations

A PLBV46 Master and a PLBV46 Slave that are connected via a Point to Point scheme will have the same Native Data Width. This avoids the need for the Master to support Conversion Cycles and for both the Master and Slave eliminates the need to support Burst length expansion.

4.3.7.2 Shared Bus Interconnect

Shared Bus interconnect generally requires additional register of inputs and outputs on the PLBV46 interface to avoid Fmax issues caused by excessive fan out and loading of shared interconnect signals. A general rule of thumb is to register all inputs and outputs from/to the PLBV46 interface however some signals cannot be registered due to required PLB transfer protocol.

4.3.8 Performance Master Normal Transfers

The following transfers are considered normal operations for the Performance Master:

- Single data beat read/write (1 to 16 bytes depending on the Master's and target Slave's native data width).
- Fixed length Burst read/write of 2 to 16 data beats of a transfer size not exceeding the native data width of the Master
  - Masters are prohibited from performing burst transfers to Baseline Slaves
  - The actual number of data beats needed to complete the request could be 2x or 4x. This is based on the ratio of the requested transfer size of the burst by the Master relative to the Slave's native data width.
- Optionally, Cacheline transfers of 4 or 8 words may be supported by Masters that are processing elements
  - The number of actual data beats required is dependent on the smaller of either the Master's or target Slave's native data width

4.3.9 Performance Master Exceptions

Should a Performance Master encounter an exception, such as a PLB_Timeout assertion or assertion of the PLB_Mrderr or PLB_Mwrerr by the target Slave, the Master is required to assert and hold the Detected Error output signal (MD_error). This assertion is maintained until the Master is reset.

Section 5

PLBV46 Bus Core

5.1 Supported Features

- 32-bit addressing
- 32, 64, or 128 bit data width
- Selectable Shared Bus or Point to Point interconnect topology
    - Point to Point optimization available for 1 master, 1 Slave configuration
    - Point to Point Topology Supports 0 Cycle Latency via arbitration removal
- Selectable address pipelining support (2 level only)
- Watchdog Timer for Address Phase Request Timeout Generation
- Dynamic Master Request Priority Based Arbitration
- Vectored Resets and Address/Qualifier Registers

5.2 Unsupported Features

- PLB Address Pipelining beyond 2 levels

5.3 PLBV46 Bus Signal Set and Usage

The PLBV46 Bus core will employ and route all defined PLBV46 signaling per the IBM CoreConnect specification except for the optional parity signals

5.4 PLBV46 Bus Parameterization

- C_PLBV46_NUM_MASTERS
- C_PLBV46_NUM_SLAVES
- C_PLBV46_MID_WIDTH
- C_PLBV46_DWIDTH
- C_ADDR_PIPELINING_TYPE
- C_PLBV46_P2P
- C_PLBV46_KEEP_TIMEOUT

Section 6

EDK Engines (platgen) Support for Parameters

The new PLBv46 bus and related IPs have a new set of requirements regarding automation needed by platgen. This section lists feature and behavior supported by platgen.

6.1 Bus Interface

The new PLBv46 will use a BUS_STD called PLBV46

6.2 Parameters

Certain parameters are required to be present on all IPs having a PLBv46 bus interface. The required parameters must be present in the MPD and don't necessarily have to exist on the HDL. If a particular 'required' parameter is not present in the HDL, then the MPD must identify such a parameter with *TYPE = NON_HDL* tag.

Information required by tools

PARAMETER C_<busifname>_NATIVE_DWIDTH (Slaves and Masters)

SUBPROPERTY GENERATE_BURSTS = TRUE | FALSE. This is for Masters BUSIF only.

Other parameters recognized and assigned values if present on the IP Core:

C_<busifname>_DWIDTH (Slaves and Masters)

C_<busifname>_MID_WIDTH (Slaves only)

C_<busifname>_NUM_MASTERS (Slaves only)

C_<busifname>_SUPPORT_BURSTS (Slaves only)

C_<busifname>_P2P (Slaves and Masters)

C_<busifname>_SMALLEST_MASTER (Slaves only)

C_<busifname>_SMALLEST_SLAVE (Masters only)

C_<busifname>_CLK_PERIOD_PS (Slaves or Masters)

6.2.1 EDK Automation of Parameter Value Assignments

EDK engines will perform the following automation regarding PLBv46 Parameters:

- DWIDTH: The assumption is that plbv46_plbv46_bridge has constant NATIVE_DWIDTH parameter. Look at NATIVE_DWIDTH parameter of each busif connected to a bus. The highest value of NATIVE_DWIDTH is set as value of DWIDTH parameter on the bus IP and all its peripherals. Note that each Bus Interface will now have a NATIVE_DWIDTH as well as a DWIDTH parameter.
- SMALLEST_MASTER/SMALLEST_SLAVE: For the SMALLEST_MASTER and SMALLEST_SLAVE parameters, it is the bridge that will do the steering/mirroring. So, tools will only look at IPs local to the bus (including the plb2plb_bridge) to set this parameter on the IPs.
- P2P: C_PLBV46_P2P parameter is set only on the bus IP and then its propagated to the master and slave connected to that bus. On the master and slave, the parameter is called C_<SPLB>_P2P and C_<MPLB>_P2P.
  - If the bus has more than 1 master or 1 slave, then this parameter is always set to 0 on the masters/slaves
  - If the bus has exactly 1 master and 1 slave, then the value of the parameter from bus is propagated to the master and slave.
- SUPPORT_BURSTS: Assuming there is no plbv46_plbv46_bridge in the system, tools will survey all Masters connected on the BUS IP to see if they have GENERATE_BURSTS sub-property set to TRUE. If any of the master busif has this value set of TRUE, then tools will set the C_<busifname>_SUPPORT_BURSTS parameter on SLAVES of the bus will be set to 1. Note that certain IPs (e.g. PLB-lite IPs like gpio) will not have any configuration parameter to support burst irrespective of the Masters connected on the bus. Such slave IPs should not have SUPPORT_BURSTS as a parameter. IPs that unconditionally support burst must still have parameter SUPPORT_BURSTS = 1 and have ASSIGNMENT = CONSTANT. This is to inform tools that this IP always supports burst. This information will be useful in configuring burst support in the bridge IP.

6.3 PLBV46 to PLBV46 Bridge Considerations

Assuming the following connection topology:

master_bus → plb2plb_bridge → slave_bus

Cascaded bridges are defined as

bus1 → plb2plb_bridge1 → bus2 → plb2plb_bridge2 → bus3

Bridges are meant for communication between the MASTERs on the master_bus and SLAVEs on the slave_bus. The SPLB busif of the bridge also has a parameter called C_SPLB_SUPPORT_BURSTS that enables the bridge to transfer burst transactions across bridges.

SUPPORT_BURSTS: It is assumed that any non-bridge master will have a fixed functionality in terms of bursts i.e. it will either always generate bursts or will never generate bursts. There will not be any parameter on a master to optionally turn-off burst generation. Thus, GENERATE_BURSTS will be a sub-property on the MASTER_BUSIF in the MPD and will not be a parameter in the MPD. The MPD syntax will be as follows:

BUS_INTERFACE BUS = MPLB, BUS_STD = PLBV46, BUS_TYPE = MASTER,
GENERATE_BURSTS = [FALSE | TRUE]

Tools will look at all masters and slaves that could communicate through a given bridge. Then, if there is at least 1 master that has GENERATE_BURSTS = TRUE and if there is at least 1 SLAVE that can support burst, then the bridge will be configured to support burst. To determine whether a slave can support burst or not, tools will use the following method:

- If slave does not have SUPPORT_BURSTS parameter, then its assumed that it does not support bursts. This means that any slave that always supports burst must also have the parameter
- If a slave has the parameter, then it is assumed that it can support bursts.

Except for SUPPORT_BURSTS, other parameter computation does not change.

6.4 Point to Point Interconnect Considerations

A PLBV46 Master and a PLBV46 Slave that are connected via a Point to Point scheme will have the same Native Data Width. This avoids the need for the Master to support Conversion Cycles and for both the Master and Slave eliminates the need to support Burst length expansion.