



PCI Express Bus Functional Model

Reference Manual

Version 1.4.5 November 2009
Copyright © PLDA 1996-2009

BFM

Reference Manual

Documentation Change History

Date	Version Number	Changes
November 2009	1.4.5	<ul style="list-style-type: none"> Updated XBFM Event interrupts
August 2009	1.4.4	<ul style="list-style-type: none"> No change
May 2009	1.4.3	<ul style="list-style-type: none"> Added PIPE disable option to Procedures/Tasks.
March 2009	1.4.2	<ul style="list-style-type: none"> Added tx_rate to Link wrapper. Updated XBFM Package Procedures/Tasks
November 2008	1.4.1	<ul style="list-style-type: none"> No change
August 2008	1.4.0	<ul style="list-style-type: none"> Added the IO/Memory space event constants: XBFM_IO_HIT, XBFM_MEM32_HIT, XBFM_MEM64_HIT. Added the transaction types: XBFM_RW, XBFM_READ, XBFM_WRITE. Added the process xbfm_wait_space_hit.
April 2008	1.3.4	<ul style="list-style-type: none"> No change
February 2008	1.3.3	<ul style="list-style-type: none"> No change
December 2007	1.3.2	<ul style="list-style-type: none"> No change
November 2007	1.3.1	<ul style="list-style-type: none"> No change
November 2007	1.3.0	<ul style="list-style-type: none"> Updated link wrapper & pipe wrapper interfaces description Update LTSSM encoding
September 2007	1.2.4	<ul style="list-style-type: none"> No change
July 2007	1.2.3	<ul style="list-style-type: none"> Added xbfm_action procedure and XBFM_Lx_STATE events
April 2007	1.2.2	<ul style="list-style-type: none"> No change
March 2007	1.2.1	<ul style="list-style-type: none"> Added xbfm_set_aspm procedure to configure active state power management Added missing xbfm_set_maxpayload procedure
January 2007	1.2	<ul style="list-style-type: none"> Added interface to connect external checkers Added xbfm_custom_tlp procedure Completion status and data can now be retrieved with xbfm_wait_id Maximum payload is user-selectable (128 bytes .. 4KBytes) Added description for external checker interface that permits you to connect a user-defined checker to the BFM Added description for XBFM_TIMEOUT Added descriptions for the following new Transaction Procedures: xbfm_dword_id, xbfm_burst_id, xbfm_custom_tlp, xbfm_custom_tlp_id, xbfm_wait_id
November 2006	1.1	The Testbench now permits configurable log file output. Added description for a new procedure, xbfm_configure_log
June 2006	1.0	First Release

Proprietary Notice

Words and logos marked with ® or ™ are registered trademarks or trademarks owned by PLDA SA. Other brands and names mentioned herein may be the trademarks of their respective owners.

Neither the whole nor any part of the information contained in, or the product described in, this document may be adapted or reproduced in any material form except with the prior written permission of the copyright holder.

The product described in this document is subject to continuous developments and improvements. All particulars of the product and its use contained in this document are given by PLDA in good faith. This document is provided "as is" with no warranties whatsoever, including any warranty of merchantability, non infringement, fitness for any particular purpose, or any warranty otherwise arising out of any proposal, specification, or sample.

This document is intended only to assist the reader in the use of the product. PLDA shall not be liable for any loss or damage arising from the use of any information in this document, or any error or omission in such information, or any incorrect use of the product. Nor shall PLDA be liable for infringement of proprietary rights relating to use of information in this document. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted herein.

Product Status

The information in this document is final content pertaining to the PLDA Bus Functional Model (BFM).

Web Address

<http://www.plda.com>

Table of Contents

List of Tables	6
List of Figures	7
Preface	8
About this Document	8
Additional Reading	8
Feedback and Contact Information	9
Chapter 1 Introduction	10
1.1 The BFM as Root Port	10
1.2 The BFM as Endpoint	10
Chapter 2 Exploring the BFM Architecture	11
2.1 Introduction	11
2.2 Functional Modules	11
2.2.1 Transactor	11
2.2.2 Monitor	11
2.2.3 PIPE	11
2.2.4 Checker	12
Chapter 3 Wrapper Implementations	14
3.1 Link Wrapper	14
3.2 PIPE Wrapper	15
Chapter 4 XBFM Package Constants and Procedures/Tasks	17
4.1 Constants	17
4.2 Procedures/Tasks	19
4.2.1 Printing Procedures/Tasks	19
4.2.1.1 <i>xbfm_print</i>	19
4.2.1.2 <i>xbfm_print_comment</i>	19
4.2.1.3 <i>xbfm_print_error</i>	19
4.2.2 Initialization Procedures/Tasks	20
4.2.2.1 <i>xbfm_set_requesterid</i>	20
4.2.2.2 <i>xbfm_init</i>	20
4.2.2.3 <i>xbfm_configure_log</i>	20
4.2.2.4 <i>xbfm_set_maxpayload</i>	21

4.2.2.5	<i>xbfm_set_aspm</i>	21
4.2.2.6	<i>xbfm_action</i>	21
4.2.3	Transaction Procedures/Tasks	22
4.2.3.1	<i>xbfm_dword</i>	22
4.2.3.2	<i>xbfm_dword_id</i>	22
4.2.3.3	<i>xbfm_burst</i>	23
4.2.3.4	<i>xbfm_burst_id</i>	23
4.2.3.5	<i>xbfm_custom_tlp</i>	24
4.2.3.6	<i>xbfm_custom_tlp_id</i>	24
4.2.3.7	<i>xbfm_set_cplstatus</i>	25
4.2.3.8	<i>xbfm_wait</i>	25
4.2.3.9	<i>xbfm_wait_id</i>	25
4.2.3.10	<i>xbfm_wait_event</i>	26
4.2.3.11	<i>xbfm_wait_linkup</i>	26
4.2.3.12	<i>xbfm_wait_space_hit</i>	26
4.2.4	Memory Access Procedures/Tasks	27
4.2.4.1	<i>xbfm_memory_write</i>	27
4.2.4.2	<i>xbfm_memory_read</i>	27
4.2.4.3	<i>xbfm_memory_compare</i>	28
4.2.4.4	<i>xbfm_memory_dump</i>	28
4.2.5	Utilities	29
4.2.5.1	<i>xbfm_buffer_fill</i>	29

List of Tables

Table 1: External checker interface	12
Table 2: Link wrapper interface	14
Table 3: PIPE wrapper interface	15
Table 4: xbfm_print parameters	19
Table 5: xbfm_print_comment parameters	19
Table 6: xbfm_print_error parameters	19
Table 7: xbfm_set_requesterid parameters	20
Table 8: xbfm_init parameters	20
Table 9: xbfm_configure_log parameters	20
Table 10: xbfm_set_maxpayload parameters	21
Table 11: xbfm_set_aspm parameters	21
Table 12: xbfm_action parameters	21
Table 13: xbfm_dword parameters	22
Table 14: xbfm_dword parameters	22
Table 15: xbfm_burst parameters	23
Table 16: xbfm_burst parameters	23
Table 17: xbfm_custom_tlp parameters	24
Table 18: xbfm_custom_tlp parameters	24
Table 19: xbfm_set_cplstatus parameters	25
Table 20: xbfm_wait parameters	25
Table 21: xbfm_wait_id parameters	25
Table 22: xbfm_wait_event parameters	26
Table 23: xbfm_wait_linkup parameters	26
Table 24: xbfm_wait_space_hit parameters	26
Table 25: xbfm_memory_write parameters	27
Table 26: xbfm_memory_read parameters	27
Table 27: xbfm_memory_compare parameters	28
Table 28: xbfm_memory_dump parameters	28
Table 29: xbfm_buffer_fill parameters	29

List of Figures

Figure 1: Configuring the BFM as a Root Port	10
Figure 2: Configuring the BFM as an Endpoint.....	10
Figure 3: BFM system architecture	11
Figure 4: Sample of the PIPE log file	12
Figure 5: Sample of the Transaction log file	12

Preface

About this Document

Intended Audience

This document has been written for design managers, system engineers, and designers of ASICs and FPGAs who are evaluating or using PLDA PCI Express products.

Scope

This document provides the complete functional description of the PLDA BFM.

Typographical Conventions

<i>italic</i>	Highlights important notes or publications
bold	Highlights interface elements.
COURIER NEW	DENOTES TEXT USED IN A CODE EXAMPLE OR A SIGNAL.

Additional Reading

This section lists additional resources from PLDA and third-parties.

PLDA periodically updates its documentation. Please contact PLDA Technical Support or check the Web site at <http://www.plda.com> for current versions.

PLDA Publications

This reference manual describes the PLDA BFM that is compatible with PCI Express-based IP core. Please refer to the Reference Manual and Getting Started guides of your particular IP core for more details.

Other Publications

Please refer to the following documents for information on specification standards:

- *PCI Express™ Base Specification Revision 2.0*
- *PCI Express™ Base Specification Revision 1.1*

Feedback and Contact Information

Feedback about this Document

PLDA welcomes comments and suggestions about this documentation. Please contact PLDA Technical Support and provide the following information:

- the title of the document
- the page number to which your comments refer
- a description of your comments

Contact information

Corporate Headquarters

PLDA
Parc club du golf - Bât. 11a
Rue Guilibert
13856 Aix-en-Provence Cedex 3 - France

Tel: USA +1 408 273 4528 - International +33 442 393 600

Fax: +33 442 394 902

Sales

For sales questions, please contact sales@plda.com.

Technical Support

For technical support questions, please contact PLDA Support at http://www.plda.com/plda_login.php using the Support Center if you have a PLDA online account.

If you don't have a PLDA account, contact http://www.plda.com/support_enquiry.php.

Chapter 1 Introduction

PLDA's PCI Express Bus Functional Model (BFM) allows you to test your PCI Express 1.1 or 2.0 design. The BFM, which emulates a PCIe environment, is connected to a Design Under Test (DUT) and supports:

- sending packets to the DUT
- receiving packets from the DUT
- monitoring bus activity
- reporting errors

1.1 The BFM as Root Port

In order to test an Endpoint or Switch upstream port design, the BFM must act as a Root Port and support:

- link training
- initialization of the DUT
- send/receive transactions to/from the DUT

The figure below illustrates a system architecture where the DUT is an Endpoint or Switch upstream port and the BFM is treated as a Root Port:

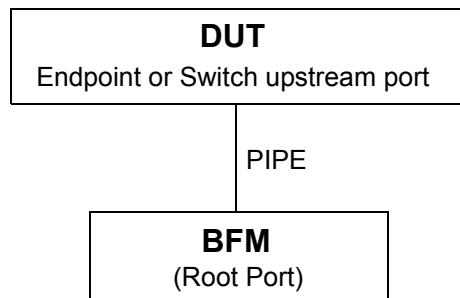


Figure 1: Configuring the BFM as a Root Port

1.2 The BFM as Endpoint

In order to test a Root Port or Switch downstream port design, the BFM must act as an Endpoint and be able to send and receive transactions to/from the DUT.

The figure below illustrates a system architecture where the DUT is a Root Port or Switch downstream port and the BFM is treated as an Endpoint:

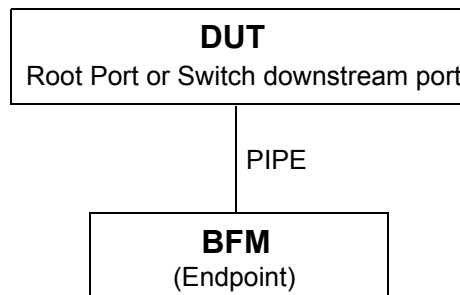


Figure 2: Configuring the BFM as an Endpoint

Chapter 2 Exploring the BFM Architecture

2.1 Introduction

PLDA's BFM includes three functional modules, as illustrated below:

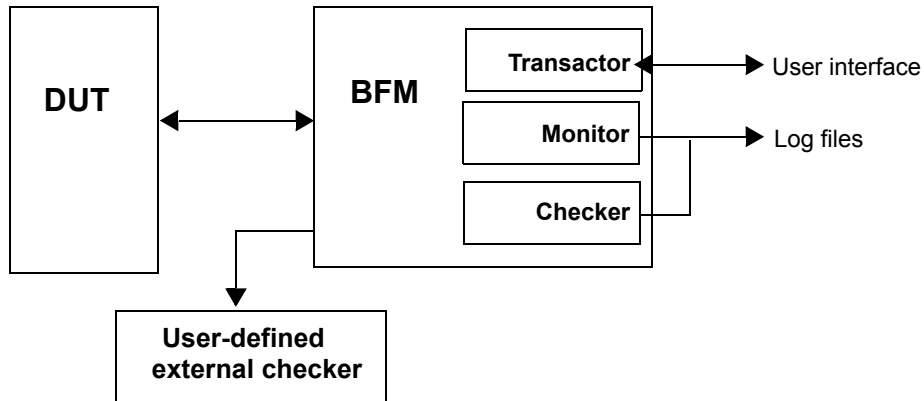


Figure 3: BFM system architecture

2.2 Functional Modules

2.2.1 Transactor

The Transactor is the heart of the BFM, responsible for sending and receiving transactions to and from the DUT. It is also responsible for:

- Implementing a memory space with a 64-bit address, a memory space with a 32-bit address, and an I/O space (each space is configurable from 4KB to 16MB)
- performing link training
- detecting events
- sending packets and performing customer-designed actions programmed with the XBFM package procedures

2.2.2 Monitor

The Monitor module surveys PCIe bus activity in both directions and logs all activity in a readable format. Information is logged at the PIPE and Transaction levels.

2.2.3 PIPE

10-bit codes are logged and special codes are replaced with their mnemonics. This log also reports LTSSM state transitions in order to help troubleshoot link training and link instability issues.

The following figure illustrates an extract from *pciebfm0_pipe.log*. You may instantiate up to four BFM's simultaneously (defined by wrapper parameter **BFM_ID**), which would lead to additional log files, for example, *pciebfm1_pipe.log* and *pciebfm2_pipe.log*.

Table 1: External checker interface

Name	Type	Description
chk_ltssm	output[4:0]	BFM current LTSSM state: 00000: detect.quiet 01110: recovery.idle 00001: detect.active 01111: L0 00010: polling.active 10000: disable 00011: polling.compliance 10001: loopback.entry 00100: polling.configuration 10010: loopback.active 00110: config.linkwidthstart 10011: loopback.exit 00111: config.linkaccept 10100: hot.reset 01000: config.lanenumaccept 10101: L0s 01001: config.lanenumwait 10110: L1.entry 01010: config.complete 10111: L1.idle 01011: config.idle 11000: L2.idle 01100: recovery.rcvlock 11001: L2.transmit.wake 01101: recovery.rcvconfig 11010: recovery.speed

Chapter 3 Wrapper Implementations

PLDA provides two wrappers for the BFM:

- Link wrapper
- PIPE wrapper

3.1 Link Wrapper

The Link wrapper binds the BFM with logic in order to provide clock, reset, and the following interfaces:

- 1 bit
- 10 bits
- 20 bits

The following table describes the Link wrapper interface:

Table 2: Link wrapper interface

Name	Type	Description
BFM_ID	Integer parameter	Unique ID for this BFM instance (0..3)
BFM_TYPE	Bit	Indicates BFM type ('0'=RP,'1'=EP)
BFM_LANES	Integer parameter	Indicates number of connected lanes (1,2, 4, or 8)
BFM_WIDTH	Integer parameter	Indicates link type (1:serial, 10: 10bits, 20: 20-bit)
IO_SIZE	Integer parameter	Specifies the size of internal IO space. Possible values range from 12 (2^{12} =4KB) to 24 (2^{24} =16MB)
MEM32_SIZE	Integer parameter	Specifies the size of internal 32-bit addressing memory space. Possible values range from 12 (2^{12} =4KB) to 24 (2^{24} =16MB)
MEM64_SIZE	Integer parameter	Specifies the size of internal 64-bit addressing memory space. Possible values range from 12 (2^{12} =4KB) to 24 (2^{24} =16MB)
clk125	Input	125 MHz clock for 2.5 Gbps mode. This clock must be switched to 250 MHz when DUT enters 5.0 Gbps mode.
clk250	Input	250 MHz clock for 2.5 Gbps mode. This clock must be switched to 500 MHz when DUT enters 5.0 Gbps mode.
rstn	Input	Active-low reset
tx_rate	Output	Indicates transmitted data rate: <ul style="list-style-type: none"> • 0 = 2.5 Gbps • 1 = 5.0 Gbps This signal must be used to control the frequency of clk125/clk250 clocks.
tx_val	Input [7..0]	TX valid bit for each lane (not used in serial mode)
tx_in0 .. tx_in7	Input [BFM_WIDTH-1:0]	TX lanes

Table 2: Link wrapper interface

Name	Type	Description
rx_val	Output [7..0]	RX valid bit for each lane (not used in serial mode)
rx_out0 .. rx_out7	Output [BFM_WIDTH-1:0]	RX lanes

3.2 PIPE Wrapper

The PIPE wrapper binds the BFM with logic in order to provide clock, reset, and the following interfaces:

- 8-bit PIPE
- 16-bit PIPE

The table below describes the PIPE wrapper interface:

Table 3: PIPE wrapper interface

Name	Type	Description
BFM_ID	Integer parameter	Unique ID for this BFM instance (0..3)
BFM_TYPE	Bit	Indicates BFM type ('0'=RP,'1'=EP)
BFM_LANES	Integer parameter	Indicates number of connected lanes (1,2, 4, or 8)
BFM_WIDTH	Integer parameter	Indicates link type (8:8-bit PIPE, 16: 16-bit PIPE)
IO_SIZE	Integer parameter	Specifies the size of internal IO space. Possible values range from 12 (2^{12} =4KB) to 24 (2^{24} =16MB)
MEM32_SIZE	Integer parameter	Specifies the size of internal 32-bit addressing memory space. Possible values range from 12 (2^{12} =4KB) to 24 (2^{24} =16MB)
MEM64_SIZE	Integer parameter	Specifies the size of internal 64-bit addressing memory space. Possible values range from 12 (2^{12} =4KB) to 24 (2^{24} =16MB)
clk125	Input	125 MHz clock in 2.5 Gbps mode. This clock must be switched to 250 MHz when DUT enters 5.0 Gbps mode.
clk250	Input	250 MHz clock in 2.5 Gbps mode. This clock must be switched to 500 MHz when DUT enters 5.0 Gbps mode.
rstn	Input	Active-low reset
rate	Input	PIPE interface signals common to all lanes. The rate signal indicates DUT data rate (0: 2.5 Gbps, 1: 5.0 Gbps). This signal must be tied to '0' if DUT supports 2.5 Gbps only.
tx_detectrx	Input	
phy_status	Output	
power_down	Input	

Table 3: PIPE wrapper interface

Name	Type	Description
tx_elecidle	Input [7:0]	PIPE interface signals for each lane.
tx_compl	Input [7:0]	
rx_polarity	Input [7:0]	
rx_elecidle	Output [7:0]	
rx_valid	Output [7:0]	
tx_data0..tx_data7	Input [BFM_WIDTH-1:0]	
tx_datak0..tx_datak7	Input [BFM_WIDTH/8-1:0]	
rx_data0..rx_data7	Output [BFM_WIDTH-1:0]	
rx_datak0..rx_datak7	Output [BFM_WIDTH/8-1:0]	
rx_status0..rx_status7	Output [2:0]	

Chapter 4 XBFM Package Constants and Procedures/Tasks

4.1 Constants

The following constants, all integers, are defined in the xbfm package:

Transaction types

- XBFM_CFGRD0: configuration read type 0
- XBFM_CFGRD1: configuration read type 1
- XBFM_IORD: IO read
- XBFM_MRD: memory read
- XBFM_MRDLK: memory read locked
- XBFM_CFGWR0: configuration write type 0
- XBFM_CFGWR1: configuration write type 1
- XBFM_IOWR: IO write
- XBFM_MWR: memory write

Completion status

- XBFM_SC: successful
- XBFM_CA: completer abort
- XBFM_UR: unsupported request
- XBFM_TIMEOUT: timeout occurred, completion was not received within 50 us

Events

- XBFM_REQ_RCVD: request received
- XBFM_CPLD_RCVD: completion with data is received
- XBFM_CPL_RCVD: completion without data received (any completion status)
- XBFM_CPLSC_RCVD: completion without data received with successful status
- XBFM_CPLUR_RCVD: completion without data received with unsupported request status
- XBFM_CPLCA_RCVD: completion without data received with completer abort status
- XBFM_INTAA_RCVD: "interrupt pin A asserted" message received
- XBFM_INTAD_RCVD: "interrupt pin A de-asserted" message received
- XBFM_INTBA_RCVD: "interrupt pin B asserted" message received
- XBFM_INTBD_RCVD: "interrupt pin B de-asserted" message received
- XBFM_INTCA_RCVD: "interrupt pin C asserted" message received
- XBFM_INTCD_RCVD: "interrupt pin C de-asserted" message received
- XBFM_INTDA_RCVD: "interrupt pin D asserted" message received
- XBFM_INTDD_RCVD: "interrupt pin D de-asserted" message received
- XBFM_PMPME_RCVD: "PME pin event" message received
- XBFM_IO_HIT: IO space received a request
- XBFM_MEM32_HIT: Request received for 32-bit address memory space
- XBFM_MEM64_HIT: Request received for 64-bit address memory space
- XBFM_L0_STATE: BFM LTSSM in L0 state
- XBFM_L0S_STATE: BFM LTSSM in L0s state

- XBFM_L1_STATE: BFM LTSSM in L1 state
- XBFM_L2_STATE: BFM LTSSM in L2 state

Spaces

- XBFM_IO: IO space
- XBFM_MEM32: Memory space with 32-bit address
- XBFM_MEM64: Memory space with 64-bit address

Log

- XBFM_LOG_CMD: log command issued
- XBFM_LOG_DATA: log transmitted / expected data
- XBFM_LOG_NOPIPE: disable PIPE logging

ASPM

- XBFM_ASPM_L0S: allow active state power management L0s state
- XBFM_ASPM_L1: allow active state power management L1 state

Actions

- XBFM_ENTER_L2: start negotiation to enter L2 state (rootport mode only)
- XBFM_SEND_PME: send PM event (endpoint mode only)

Transaction Types

- XBFM_RW: read or write request
- XBFM_READ: read request only
- XBFM_WRITE: write request only

4.2 Procedures/Tasks

The following section describes the Procedures/Tasks in the XBFM Package.

Note: These functions are known as Procedures in VHDL and Tasks in Verilog.

4.2.1 Printing Procedures/Tasks

4.2.1.1 xbfm_print

Prints a line to a log file.

Table 4: xbfm_print parameters

Name	Type	Description
bfmid	Integer	Indicates which BFM is targeted (0..3)
message	Input string	Message string, up to 100 characters

4.2.1.2 xbfm_print_comment

This Procedure/Task prints a message to a log file. This is useful for commenting a simulation and making log files more readable.

Table 5: xbfm_print_comment parameters

Name	Type	Description
bfmid	Integer	Indicates which BFM is targeted (0..3)
message	Input string	Message string, up to 100 characters

4.2.1.3 xbfm_print_error

This Procedure/Task prints an error message, with a time stamp, to a log file. This is useful for reporting errors that are not fatal and that do not require stopping a simulation.

Table 6: xbfm_print_error parameters

Name	Type	Description
bfmid	Integer	Indicates which BFM is targeted (0..3)
message	Input string	Message string, up to 100 characters

4.2.2 Initialization Procedures/Tasks

4.2.2.1 xbfm_set_requesterid

This Procedure/Task changes the BFM requester ID. The BFM will use this value as its requester ID when issuing transactions. The default requester ID for the BFM is 0000h; it is normally not necessary to change this ID.

Table 7: xbfm_set_requesterid parameters

Name	Type	Description
bfmid	Integer	Indicates which BFM is targeted (0..3), VHDL only
id	Input [15:0]	<ul style="list-style-type: none"> • id[15:8]: is bus number • id[7:3] is device number • id[2:0] is function number

4.2.2.2 xbfm_init

This Procedure/Task, which must be called at the beginning of a simulation, initializes a BFM and sets the base addresses for I/O and memory address ranges.

Table 8: xbfm_init parameters

Name	Type	Description
bfmid	Integer	Indicates which BFM is targeted (0..3), VHDL only
io_baseaddr	Input [31:0]	Specify base address for IO space (0=not used)
mem32_baseaddr	Input [31:0]	Specify base address for MEM32 space (0=not used)
mem64_baseaddr	Input [63:0]	Specify base address for MEM64 space (0=not used)

4.2.2.3 xbfm_configure_log

This Procedure/Task is used to configure log files. A value of 0, the default value, disables all options. You may implement several options at once by adding the desired values.

Table 9: xbfm_configure_log parameters

Name	Type	Description
bfmid	Integer	Indicates which BFM is targeted (0..3), VHDL only
value	Integer	Indicates log file options <ul style="list-style-type: none"> • XBFM_LOG_CMD: log command issued • XBFM_LOG_DATA: log transmitted / expected data • XBFM_LOG_NOPIPE: disable PIPE logging

4.2.2.4 xbfm_set_maxpayload

This Procedure/Task is used to configure the maximum payload size of the testbench.

Table 10: xbfm_set_maxpayload parameters

Name	Type	Description
bfmid	Integer	Indicates which BFM is targeted (0..3), VHDL only
value	Integer	Indicates maximum payload in unit of bytes. Allowed values are : 128,256,512,1024,2048,4096

4.2.2.5 xbfm_set_aspm

This Procedure/Task is used to configure active state power management. A support value of 0 disables ASPM. You may implement several options by adding the desired values.

Table 11: xbfm_set_aspm parameters

Name	Type	Description
bfmid	Integer	Indicates which BFM is targeted (0..3), VHDL only
support	Integer	Indicates log file options <ul style="list-style-type: none"> • XBFM_ASPM_L0S : allow ASPM L0s • XBFM_ASPM_L1 : allow ASPM L1
delay	Integer	Indicates the idle time delay to automatically enter low power state (4...31) in steps of 256 ns.

4.2.2.6 xbfm_action

This Procedure/Task directs the BFM to execute special actions.

Table 12: xbfm_action parameters

Name	Type	Description
bfmid	Integer	Indicates which BFM is targeted (0..3), VHDL only
action	Integer	Indicates action to take <ul style="list-style-type: none"> • XBFM_ENTER_L2: start negotiation to enter L2 state (rootport mode only) • XBFM_SEND_PME: send PM event (endpoint mode only)

4.2.3 Transaction Procedures/Tasks

4.2.3.1 xbfm_dword

This Procedure/Task directs the BFM to send a 1 DW request. The transfer byte count is always four bytes and the address is 32-bit only.

Table 13: xbfm_dword parameters

Name	Type	Description
bfmid	Integer	Indicates which BFM is targeted (0..3), VHDL only
command	Integer	Indicates the type of request, supported values are: <ul style="list-style-type: none"> • XBFM_CFGRD0: configuration read type 0 • XBFM_CFGRD1: configuration read type 1 • XBFM_IORD: IO read • XBFM_MRD: memory read • XBFM_CFGWR0: configuration write type 0 • XBFM_CFGWR1: configuration write type 1 • XBFM_IOWR: IO write • XBFM_MWR: memory write
address	Input [31:0]	Specifies target address: this address must be DW-aligned (bits [1:0]=00)
be	Input [3:0]	Byte enables: any byte-enable value can be supplied. Default is 1111b which indicates that all bytes must be transferred.
data	Input [31:0]	<ul style="list-style-type: none"> • Write requests: indicates the value to write. • Read requests: if a valid value is supplied, then the XBFM will automatically check that it matches read value. Use XXXXXXXXh in order to disable data checking.

4.2.3.2 xbfm_dword_id

This Procedure/Task is the same as xbfm_dword except for an additional parameter that indicates the ID of the issued transaction. This ID can be used to track transaction with xbfm_wait_id..

Table 14: xbfm_dword parameters

Name	Type	Description
bfmid	Integer	Indicates which BFM is targeted (0..3), VHDL only
command	Integer	Indicates the type of request, supported values are: <ul style="list-style-type: none"> • XBFM_CFGRD0: configuration read type 0 • XBFM_CFGRD1: configuration read type 1 • XBFM_IORD: IO read • XBFM_MRD: memory read • XBFM_CFGWR0: configuration write type 0 • XBFM_CFGWR1: configuration write type 1 • XBFM_IOWR: IO write • XBFM_MWR: memory write
address	Input [31:0]	Specifies target address: this address must be DW-aligned (bits [1:0]=00)
be	Input [3:0]	Byte enables: any byte-enable value can be supplied. Default is 1111b which indicates that all bytes must be transferred.

Table 14: xbfm_dword parameters

Name	Type	Description
data	Input [31:0]	<ul style="list-style-type: none"> Write requests: indicates the value to write. Read requests: if a valid value is supplied, then the XBFM will automatically check that it matches read value. Use XXXXXXXXh in order to disable data checking.
id	output	ID of issued transaction.

4.2.3.3 xbfm_burst

This Procedure/Task directs the BFM to send a burst request. See the memory_write Procedure/Task description for instructions about how to prepare a data buffer.

Table 15: xbfm_burst parameters

Name	Type	Description
bfmid	Integer	Indicates which BFM is targeted (0..3), VHDL only
command	Integer	Indicates the type of request. Supported values are: <ul style="list-style-type: none"> XBFM_MRD: memory read XBFM_MRDLK: memory read locked XBFM_MWR: memory write
address	Input [63:0]	Specifies the target address, which can start on any byte
bytecount	Integer	Total transfer byte count: any value from 1 to 4096 bytes can be specified.
data	Input [*][31:0]	<ul style="list-style-type: none"> Write requests: indicates the values to write. Read requests: if a valid data are supplied then the XBFM will automatically check that these match read values. Use XXXXXXXXh for first data word in order to disable data checking.
tc	Input [2:0]	Traffic class: any value from 000b to 111b is allowed; default is 000b.
attr	Input [1:0]	Allows you to specify values for relaxed-ordering (RO) and no-snoop (NS) bits. Any value is allowed. Default is 00b.

4.2.3.4 xbfm_burst_id

This Procedure/Task is the same as xbfm_burst except that it outputs an additional parameter that indicates the ID of the issued transaction. This ID can be use to track the transaction using xbfm_wait_id.

Table 16: xbfm_burst parameters

Name	Type	Description
bfmid	Integer	Indicates which BFM is targeted (0..3), VHDL only
command	Integer	Indicates the type of request. Supported values are: <ul style="list-style-type: none"> XBFM_MRD: memory read XBFM_MRDLK: memory read locked XBFM_MWR: memory write
address	Input [63:0]	Specifies the target address, which can start on any byte
bytecount	Integer	Total transfer byte count: any value from 1 to 4096 bytes can be specified.

Table 16: xbfm_burst parameters

Name	Type	Description
data	Input [*][31:0]	<ul style="list-style-type: none"> Write requests: indicates the values to write. Read requests: if a valid data are supplied then the XBFM will automatically check that these match read values. Use XXXXXXXXh for first data word in order to disable data checking.
tc	Input [2:0]	Traffic class: any value from 000b to 111b is allowed; default is 000b.
attr	Input [1:0]	Allows you to specify values for relaxed-ordering (RO) and no-snoop (NS) bits. Any value is allowed. Default is 00b.
id	output	ID of issued transaction.

4.2.3.5 xbfm_custom_tlp

This Procedure/Task directs the BFM to send a TLP with a user-defined header.

Table 17: xbfm_custom_tlp parameters

Name	Type	Description
bfmid	integer	Indicates the target BFM (0..3), VHDL only
header	input (127:0)	Header of TLP: byte 0 of header are bits [127:120],... byte 15 of header are bits [7:0] Note that: <ul style="list-style-type: none"> you must tie bits [31:0] to 0s if a 3 DW header is specified bits [95:80] must be 0s as well and are automatically replaced with the BFM requester ID
dwordcount	integer	Number of DW of data
data	input [*][31:0]	Data buffer to hold up to 4096 bytes

4.2.3.6 xbfm_custom_tlp_id

This Procedure/Task is the same as xbfm_custom_tlp, except that it outputs an additional parameter that indicates the ID of the issued transaction. This ID is used to track a transaction with xbfm_wait_id.

Table 18: xbfm_custom_tlp parameters

Name	Type	Description
bfmid	integer	Indicates the target BFM (0..3), VHDL only
header	input (127:0)	Header of TLP: byte 0 of header are bits [127:120],... byte 15 of header are bits [7:0] Note that: <ul style="list-style-type: none"> you must tie bits [31:0] to 0s if a 3 DW header is specified bits [95:80] must be 0s as well and are automatically replaced with the BFM requester ID
dwordcount	integer	Number of DW of data
data	input [*][31:0]	Data buffer to hold up to 4096 bytes.
id	output	ID of issued transaction.

4.2.3.7 xbfm_set_cplstatus

This Procedure/Task changes the status of subsequent completions sent by BFM.

Table 19: xbfm_set_cplstatus parameters

Name	Type	Description
bfmid	integer	Indicates the target BFM (0..3), VHDL only
status	Integer	Indicates the completion status to return, supported values are: <ul style="list-style-type: none"> • XBFM_SC: successful • XBFM_CA: completer abort • XBFM_UR: unsupported request
count	Integer	Number of completions to return with specified status code.

4.2.3.8 xbfm_wait

This Procedure/Task waits until the BFM has finished processing outstanding requests. If outstanding requests require completions, then this Procedure/Task will wait until all completions have been received.

Table 20: xbfm_wait parameters

Name	Type	Description
bfmid	integer	Indicates the target BFM (0..3), VHDL only

4.2.3.9 xbfm_wait_id

This Procedure/Task returns the transaction status and received data (if any) at the end of a transaction transmitted with xbfm_dword_id, xbfm_burst_id, or sbfm_custom_tlp_id.

Table 21: xbfm_wait_id parameters

Name	Type	Description
bfmid	integer	Indicates the target BFM (0..3), VHDL only
trnid	integer	Transaction ID
status	output integer	Indicates the completion status of a transaction. Possible values are: <ul style="list-style-type: none"> • XBFM_SC: transaction successful • XBFM_CA: completer abort received • XBFM_UR: unsupported request received • XBFM_TIMEOUT: timeout occurred, completion for transaction was not received
data	output [*][31:0]	Data buffer to hold up to 4096 bytes of received data (if any)

4.2.3.10 xbfm_wait_event

This Procedure/Task waits until the BFM signals a specific event. Note that execution of the calling process will be blocked if the specified event is not received.

Table 22: xbfm_wait_event parameters

Name	Type	Description
bfmid	Integer	Indicate which BFM is targeted (0..3), VHDL only
event_id	Integer	Indicates which event is expected. The value of this parameter can be any one of the events listed in Section 4.1 .

4.2.3.11 xbfm_wait_linkup

This Procedure/Task waits until link training is completed.

Table 23: xbfm_wait_linkup parameters

Name	Type	Description
bfmid	Integer	Indicates which BFM is targeted (0..3), VHDL only

4.2.3.12 xbfm_wait_space_hit

This Procedure/Task waits until a request corresponding to a specific address range in BFM memory spaces is received.

Table 24: xbfm_wait_space_hit parameters

Name	Type	Description
bfmid	Integer	Indicates which BFM is targeted (0..3), VHDL only
space	Integer	Indicates which XBFM memory space is targeted. Valid values are: <ul style="list-style-type: none"> • XBFM_IO: IO space • XBFM_MEM32: 32-bit addressing memory space • XBFM_MEM64: 64-bit addressing memory space
offsetl	Input[31:0]	Specifies the lowest offset of detection range within memory space.
offseth	Input[31:0]	Specifies the highest offset of detection range within memory space.
rw	Integer	Indicates which type of request will be detected. Valid values are: <ul style="list-style-type: none"> • XBFM_RW • XBFM_READ • XBFM_WRITE

4.2.4 Memory Access Procedures/Tasks

4.2.4.1 xbfm_memory_write

This Procedure/Task writes the content of a data buffer to a BFM memory space starting from a specified address.

Table 25: xbfm_memory_write parameters

Name	Type	Description
bfmid	Integer	Indicates which BFM is targeted (0..3), VHDL only
space	Integer	Specifies which XBFM memory space is targeted. Valid values are: <ul style="list-style-type: none"> • XBFM_IO: IO space • XBFM_MEM32: 32-bit addressing memory space • XBFM_MEM64: 64-bit addressing memory space
offset	Input[31:0]	Specifies at which offset within the memory space data write must start. This value must be DW aligned (bits [1:0]=00)
num_dword	Integer	Number of DW to write
data	Input [*][31:0]	Data buffer to hold up to 4096 bytes

4.2.4.2 xbfm_memory_read

This Procedure/Task reads the content of a BFM memory space starting from a specified address.

Table 26: xbfm_memory_read parameters

Name	Type	Description
bfmid	Integer	Indicates which BFM is targeted (0..3), VHDL only
space	Integer	Specifies which XBFM memory space is targeted. Valid values are: <ul style="list-style-type: none"> • XBFM_IO: IO space • XBFM_MEM32: 32-bit addressing memory space • XBFM_MEM64: 64-bit addressing memory space
offset	Input [31:0]	Specifies at which offset within the memory space data read must start. This value must be DW aligned (bits [1:0]=00)
num_dword	Integer	Number of DW to read
data	Output [*][31:0]	Data buffer to hold up to 4096 bytes

4.2.4.3 xbfm_memory_compare

This Procedure/Task compares the content of a data buffer with data from a BFM memory space starting from a specified address. Mismatches are reported in a log file.

Table 27: xbfm_memory_compare parameters

Name	Type	Description
bfmid	Integer	Indicates which BFM is targeted (0..3), VHDL only
space	Integer	Specifies which XBFM memory space is targeted. Valid values are: <ul style="list-style-type: none"> • XBFM_IO: IO space • XBFM_MEM32: 32-bit addressing memory space • XBFM_MEM64: 64-bit addressing memory space
offset	Input	Specifies at which offset within the memory space data compare must start. This value must be DW aligned (bits [1:0]=00)
num_dword	Integer	Number of DW to compare
data	Input [*][31:0]	Data buffer to hold up to 4096 bytes
fbe	Input [3:0]	Specifies the byte enables for first DW of comparison. Default value is 1111b that indicates that all bytes must be compared.
lbe	Input [3:0]	Specifies the byte enables for last DW of comparison. Default value is 1111b that indicates that all bytes must be compared

4.2.4.4 xbfm_memory_dump

This Procedure/Task dumps the content of a memory space to a file.

Table 28: xbfm_memory_dump parameters

Name	Type	Description
bfmid	Integer	Indicates which BFM is targeted (0..3), VHDL only
status	Integer	Specifies which XBFM memory space is targeted. Valid values are: <ul style="list-style-type: none"> • XBFM_IO: IO space • XBFM_MEM32: 32-bit addressing memory space • XBFM_MEM64: 64-bit addressing memory space
file	String	File name, up to 100 characters

4.2.5 Utilities

4.2.5.1 xbfm_buffer_fill

This Procedure/Task fills a data buffer with a data ramp (00h, 01h, 02h..., FEh, FFh). This is useful for preparing data for a burst transfer.

Table 29: xbfm_buffer_fill parameters

Name	Type	Description
num_dword	Integer	Number of DWs to fill
data	Output [*][31:0]	Data buffer to hold up to 4096 bytes