

Introduction

The Advanced Microcontroller Bus Architecture (AMBA®) Advanced eXtensible Interface (AXI4) to Processor Local Bus (PLB v4.6) Bridge translates AXI transactions into PLBv46 transactions. It functions as 32/64-bit slave on AXI4 and 32/64-bit master on the PLB.

Features

The Xilinx AXI to PLBv46 Bridge is a soft IP core with the following features:

- Support AXI4 & PLB v4.6 (Xilinx simplification)
- Supports 1:1 (AXI:PLB) synchronous clock ratio
- Support 32-bit address on AXI & PLB interfaces
- Supports 32/64-bit data buses on AXI & PLB interfaces (1:1 ratio)
- Supports Write and read data buffering

AXI4 Slave interface

- Supports Configurable AXI4 Interface Categories
 - Control (AXI4- Lite) Interface
 - Read/Write Interface
 - Read-only Interface
 - Write-only Interface
- Additional control interface to access internal registers of the design
- Supports INCR bursts of 1 to 256
- Supports bursts of 1-16 for FIXED type transfer
- Supports burst of 2, 4, 8 and 16 for WRAP type transfers
- Configurable support for narrow transfers
- Supports unaligned transactions
- Early response for bufferable write transfer

LogiCORE IP Facts Table					
Core Specifics					
Supported Device Family ⁽¹⁾	Artix-7, Virtex-7, Kintex-7, Spartan-6 ⁽²⁾ , Virtex-6 ⁽³⁾				
Supported User Interfaces	AXI4, AXI4-Lite, PLBv46				
	Resources				Frequency
	LUTs	FFs	Slices	Block RAMs	Max. Freq.
	Refer to Table 11 and Table 12			0	
Provided with Core					
Documentation	Product Specification				
Design Files	VHDL				
Example Design	Not Provided				
Test Bench	Not Provided				
Constraints File	Not Provided				
Simulation Model	None				
Tested Design Tools					
Design Entry Tools	XPS 13.2				
Simulation ⁽⁴⁾	Mentor Graphics ModelSim				
Synthesis Tools	XST 13.2				
Support					
Provided by Xilinx, Inc.					

Notes:

1. For a listing of supported devices, see the [release notes](#) for this core.
2. For more information on the Spartan®-6 devices, see the [Spartan-6 Family Overview \[Ref 3\]](#)
3. For more information on the Virtex®-6 devices, see the [Virtex-6 Family Overview \[Ref 4\]](#)
4. For the supported versions of the tools, see the [ISE Design Suite 13: Release Notes Guide](#).

Features (continued)

- Debug register for error/timeout condition for bufferable write transfer
- Configurable (max two) number of pipelined read/write addresses
- Interrupt generation for write data strobes null
- Interrupt generation for partial data strobes except first and last data beat
- Supports simultaneous read and write operations

PLBv46 Master interface

- Configurable (max two) number of pipelined read/write address
- Supports Xilinx simplified PLBv46 protocol
 - Single transfers of 1 to 4/8 bytes
 - Fixed length of 2 to 16 data beat
 - Cacheline transactions of line size 4 & 8
- Address pipelining for one read and one write
- Supports simultaneous read and write operations
- Supports 32, 64, and 128-bit PLBv46 data bus widths with required data mirroring

Functional Description

Overview

A block diagram for the AXI to PLB bridge is shown in [Figure 1](#). The PORT-2 shown in [Figure 1](#) is valid only when `C_EN_DEBUG_REG=1`, `C_S_AXI_PROTOCOL="AXI4"`, and `C_S_AXI_SUPPORTS_WRITE=1`. The more detailed view for the configuration when `C_S_AXI_PROTOCOL="AXI4"` AND `C_S_AXI_SUPPORTS_WRITE=1` and `C_S_AXI_SUPPORTS_READ=1` is shown in [Figure 2](#).

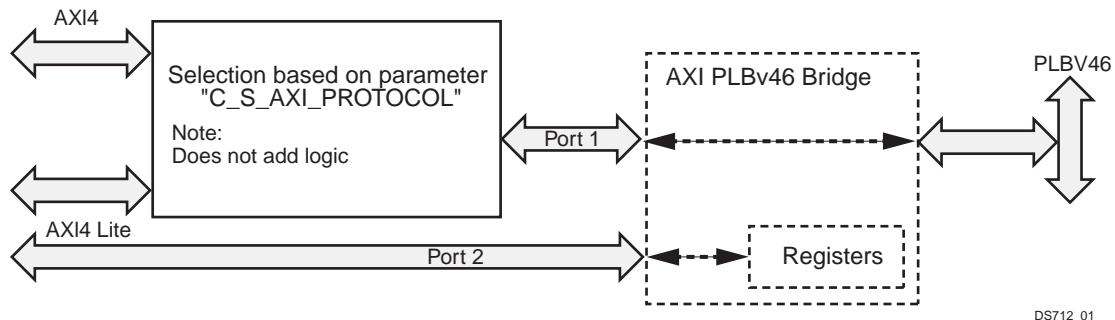
The AXI data bus width is a 32/64-bit and the PLBv46 master is a 32/64-bit device (i.e., `C_MPLB_NATIVE_DWIDTH = 32/64`). PLBv46 data bus widths of 32-bit, 64-bit and 128-bit are supported with the AXI to PLBv46 bridge performing data mirroring that is required.

AXI transactions are received on the AXI slave interface, then translated to PLBv46 transactions on the PLBv46 bus through PLBv46 master interface. Both read data and write data are buffered (when `C_S_AXI_PROTOCOL="AXI4"`) in the bridge, because of the mismatch of AXI and PLBv46 protocols where AXI allows the master to throttle data flow, but PLBv46 protocol does not allow PLB masters to throttle data flow.

The write data input from the AXI port is buffered in the bridge before the PLBv46 write transaction is initiated.

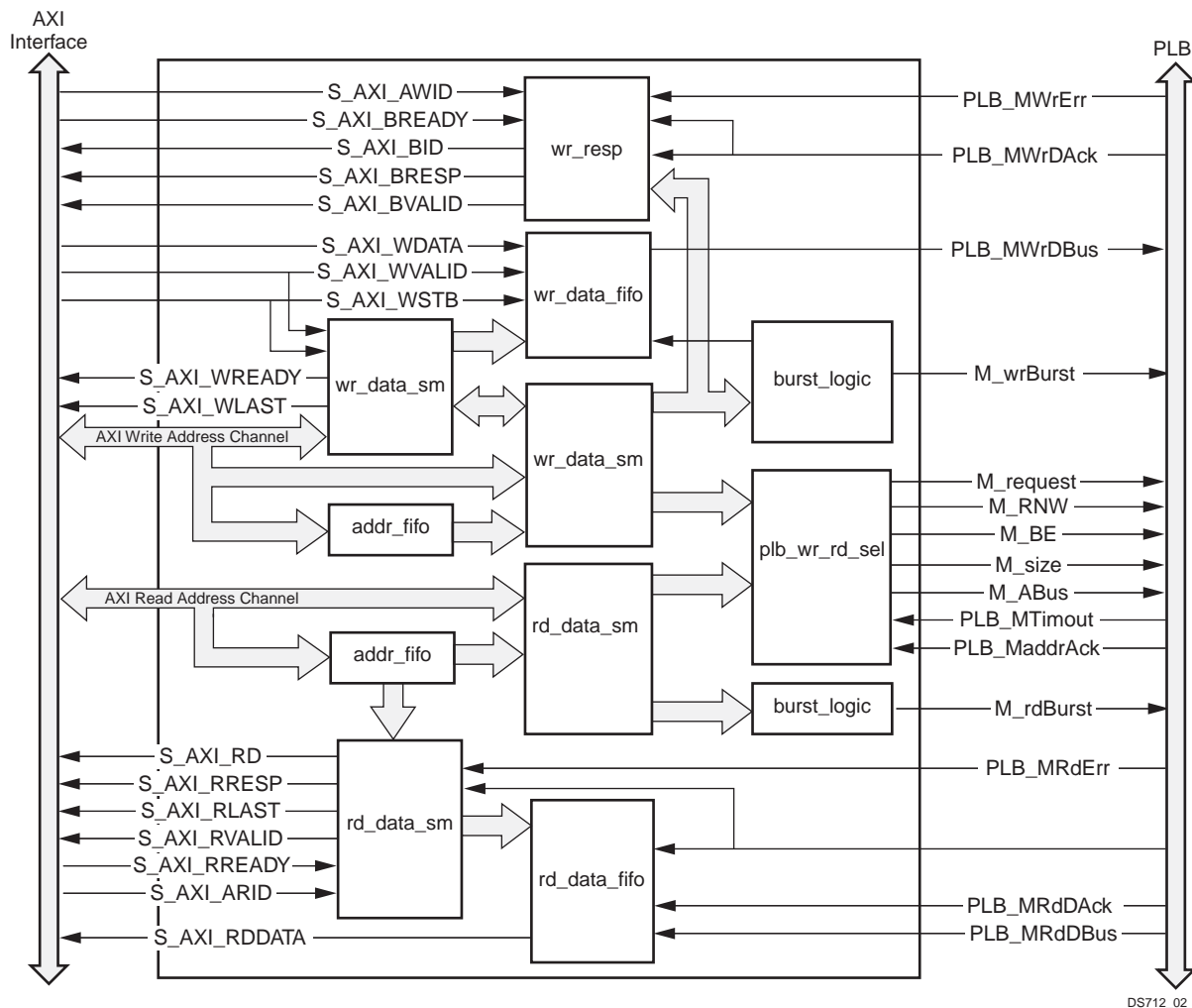
Read and write data buffer of depth 32x32/64x32 is implemented to hold the data for two PLB transfers of highest (16) burst length.

Simultaneous read and write operations from AXI to PLB are supported.



DS712_01

Figure 1: AXI PLBv46 Bridge Block Diagram



DS712_02

Figure 2: AXI PLBv46 Bridge Block Diagram (C_S_AXI_PROTOCOL="AXI4")

AXI4-Lite - PLBv46 Bridge

This module is not shown in the [Figure 2](#) because it is implemented only when the AXI interface is AXI4-Lite, for example, parameter `C_S_AXI_PROTOCOL="AXI4LITE"`.

This module converts all AXI4-Lite transactions to the PLBv46 transactions.

Xfer Qual Gen (xfer_qual_gen)

Implemented only for AXI4 interface i.e. parameter `C_S_AXI_PROTOCOL="AXI4"`.

This module (not shown in block diagram) is used to decode the both read and write AXI address channel.

Write Data State Machine (wr_data_sm)

Implemented only for AXI4 interface and not read only i.e. parameters `C_S_AXI_PROTOCOL="AXI4"` and `C_S_AXI_SUPPORTS_WRITE=1`.

AXI can generate INCR, narrow transfers that are converted to word width to get the better throughput. AXI can also generate the WRAP transfers where address is not align to the WRAP boundary. The sequence of this data needs to be changed (address aligned) on PLB v46. This module generates the control signals for `wr_data_fifo`.

FIFO FWFT 2 Deep (fifo_fwft_2deep)

The AXI to PLBv46 Bridge design supports the deasserted data strobes (`S_AXI_WSTB`) in first and last data beat only. To hold the first and last data strobe information for two transaction two; FIFO of depth is used one for each data strobe.

This is two deep first word fall through FIFO (not shown in block diagram) is implemented using registers. This is used in `wr_data_sm` to store the `first_ds`, `last_ds`.

Address FIFOs

FIFOs of depth two are used to register the write and read address channel signals.

Write Address State Machine (wr_addr_sm)

Implemented only for AXI4 interface and not read only, that is, parameters `C_S_AXI_PROTOCOL="AXI4"` and `C_S_AXI_SUPPORTS_WRITE=1`.

This module will generate the `wr_request`, `be`, `size`, `burst` for `plb_wr_rd_sel` and `burst_logic` modules. The address will not be initiated on PLB till the last data (`S_AXI_WLAST`) from the AXI for that transfer is received.

Write Address Generation (wr_addr_gen)

Implemented only for AXI4 interface and not read only, that is, parameters `C_S_AXI_PROTOCOL="AXI4"` and `C_S_AXI_SUPPORTS_WRITE=1`.

This module (not shown in block diagram) is used to generate the write address for the PLB transfer.

Write Data FIFO (wr_data_fifo)

Implemented only for AXI4 interface and not read only, that is, parameters `C_S_AXI_PROTOCOL="AXI4"` and `C_S_AXI_SUPPORTS_WRITE=1`.

This is 32x32/64x32 FIFO used to store the write data generated from AXI and will be read on `PLB_MWrDack`.

Write Response (wr_resp)

Implemented only for AXI4 interface and not read only, that is, parameters C_S_AXI_PROTOCOL="AXI4" and C_S_AXI_SUPPORTS_WRITE=1.

This module will generate the response for the write transfer. This also has FIFO (2 deep) to store the transaction IDs generated from AXI.

Read Address State Machine (rd_addr_sm)

Implemented only for AXI4 interface and not write only, that is, parameters C_S_AXI_PROTOCOL="AXI4" and C_S_AXI_SUPPORTS_READ=1.

This converts the AXI address to the PLB address for the read transfers. This also does the necessary conversion of burst length in case of narrow transfers generated from AXI to word transfer on PLB.

Read Data FIFO (rd_data_fifo)

Implemented only for AXI4 interface and not write only, that is, parameters C_S_AXI_PROTOCOL="AXI4" and C_S_AXI_SUPPORTS_READ=1.

This is 32x32 FIFO used to store the read data generated from PLB and will be read on S_AXI_RREADY.

Read Data State Machine (rd_data_sm)

Implemented only for AXI4 interface and not write only, that is, parameters C_S_AXI_PROTOCOL="AXI4" and C_S_AXI_SUPPORTS_READ=1.

This will read the data from rd_data_fifo and send to AXI along with S_AXI_RVALID and S_AXI_RLAST. This also generates the read response for AXI.

Burst Logic (burst_logic)

Implemented only for AXI4 interface, that is, parameters C_S_AXI_PROTOCOL = "AXI4".

This is used to generate the M_wrBurst and M_rdBurst signals for PLB.

PLB Wr Rd Select (plb_wr_rd_sel)

Implemented only for AXI4 interface and when supports both read and write i.e. parameters C_S_AXI_PROTOCOL = "AXI4" and C_S_AXI_SUPPORTS_WRITE=1 and C_S_AXI_SUPPORTS_READ=1.

This is used to generate the final address qualifiers on PLB. Default read is always high priority.

Design Parameters

Table 1 shows the design parameters of the AXI to PLBv46 Bridge.

Inferred Parameters

In addition to the parameters listed in Table 1, there are also parameters that are inferred for each AXI interface in the EDK tools. Through the design, these EDK-inferred parameters control the behavior of the AXI Interconnect. For a complete list of the interconnect settings related to the AXI interface, see the DS768, *AXI Interconnect IP Data Sheet*.

WRITER NOTE: Don't we need to add the new devices in G1?

Table 1: Design Parameters

Generic	Feature/Description	Parameter Name	Allowable Values	Default Values	VHDL Type
System Parameters					
G1	Target FPGA family	C_FAMILY	artix7, virtex7, kintex7, virtex6, spartan6	virtex6	string
Bridge Interface Parameters					
G2	Enable byte swap for AXI4	C_EN_BYTE_SWAP	0,1	0	integer
G3	Implement Debug register	C_EN_DEBUG_REG ⁽¹⁾	0,1	0	integer
G4	AXI Protocol to connected master	C_S_AXI_PROTOCOL	"AXI4LITE", "AXI4"	"AXI4LITE"	string
G5	Implement Narrow transfer support	C_S_AXI_SUPPORTS_NARROW_BURST ⁽¹⁾	0,1	1	integer
G6	Implement Exclusive access support	C_S_AXI_SUPPORTS_EXCL_ACCE SS ⁽¹⁾	0	0	integer
G7	AXI Identification tag width	C_S_AXI_ID_WIDTH ⁽¹⁾	1-16	4	integer
G8	AXI most significant address bus width	C_S_AXI_ADDR_WIDTH	32	32	integer
G9	AXI data bus width	C_S_AXI_DATA_WIDTH	32, 64	32	integer
G10	Indicates whether write channel is included in the design	C_S_AXI_SUPPORTS_WRITE ⁽¹⁾	0, 1	1	integer
G11	Indicates whether read channel is included in the design	C_S_AXI_SUPPORTS_READ ⁽¹⁾	0, 1	1	integer
G12	Maximum number of active write transactions that slave can accept	C_S_AXI_WRITE_ACCEPTANCE ⁽¹⁾	1, 2	1	integer
G13	Maximum number of active read transactions that slave can accept	C_S_AXI_READ_ACCEPTANCE ⁽¹⁾	1, 2	1	integer
G14	Indicates if slave supports barrier transactions	C_S_AXI_SUPPORTS_BARRIERS ⁽¹⁾	0	0	integer

Table 1: Design Parameters

Generic	Feature/Description	Parameter Name	Allowable Values	Default Values	VHDL Type
G15	Bridge Base Address	C_S_AXI_RINGx_BASEADDR ⁽²⁾	Valid Address	All ONEs	std_logic_vector
G16	Bridge High Address	C_S_AXI_RINGx_HIGHADDR ⁽²⁾	Valid Address	All ZEROs	std_logic_vector
G16	Defines bridge address ranges	C_S_AXI_NUM_ADDR_RANGES	1-4	1	integer
Bridge Register Interface Parameters ⁽³⁾					
G17	AXI most significant address bus width	C_S_AXI_CTRL_ADDR_WIDTH ⁽³⁾	32	32	integer
G18	AXI data bus width	C_S_AXI_CTRL_DATA_WIDTH ⁽³⁾	32	32	integer
G19	Bridge register interface base address	C_S_AXI_CTRL_BASEADDR ^(2,3)	Valid Address	All ONEs	std_logic_vector
G20	Bridge register interface high address	C_S_AXI_CTRL_HIGHADDR ^(2,3)	Valid Address	All ZEROs	std_logic_vector
PLB Parameters					
G21	PLB least significant address bus width	C_MPLB_AWIDTH	32 ⁽⁴⁾	32	integer
G22	PLB data width	C_MPLB_DWIDTH	32, 64, 128	32	integer
G23	Native width of the master Data Bus	C_MPLB_NATIVE_DWIDTH	32, 64 ⁽⁵⁾	32	integer
G24	Data width of the smallest slave that can talk to AXI PLBv46 bridge	C_MPLB_SMALLEST_SLAVE	32, 64	32	integer
G25	Define number of address pipelines supported on PLB	C_PLB_ADDRESS_PIPELINE	0 0-No address pipeline	0	integer

Notes:

- Valid only when C_S_AXI_PROTOCOL="AXI4".
- User must assign a valid address. The bridge will have address ranges based on parameter C_S_AXI_NUM_ADDR_RANGES.
- Valid only when C_S_AXI_PROTOCOL="AXI4" and C_EN_DEBUG_REG=1.
- Same as C_S_AXI_ADDR_WIDTH/C_S_AXI_CTRL_ADDR_WIDTH
- Same as C_S_AXI_DATA_WIDTH

I/O Signals

Table 2 shows the IO signals of the AXI to PLBv46 Bridge.

Table 2: IO Signal Description

Port	Signal Name	Interface	I/O	Initial State	Description
AXI Bridge Interface					
AXI Write Address Channel Signals					
P1	S_AXI_AWID[C_S_AXI_ID_WIDTH-1:0] ⁽¹⁾	AXI4	I	-	Write address ID. This signal is the identification tag for the write address group of signals
P2	S_AXI_AWADDR[C_S_AXI_ADDR_WIDTH-1:0]	AXI4/AXI4 LITE	I	-	AXI Write address. The write address bus gives the address of the first transfer in a write burst transaction

Table 2: IO Signal Description (Cont'd)

Port	Signal Name	Interface	I/O	Initial State	Description
P3	S_AXI_AWLEN[7:0]	AXI4	I	-	Burst length. This signal gives the exact number of transfers in a burst "00000000" - "11111111" indicates Burst Length 1 - 256
P4	S_AXI_AWSIZE[2:0] ⁽¹⁾	AXI4	I	-	Burst size. This signal indicates the size of each transfer in the burst. "000" - 1 Byte "001" - 2 byte (Half word) "010" - 4 byte (Word) "011" - 8 byte (Double Word) others - NA (up to 128 bytes)
P5	S_AXI_AWBURST[1:0] ⁽¹⁾	AXI4	I	-	Burst type. This signal coupled with the size information, details how the address for each transfer within the burst is calculated "00" - FIXED "01" - INCR "10" - WRAP "11" - Reserved
P6	S_AXI_AWCACHE[4:0] ⁽¹⁾	AXI4	I	-	Cache type. This signal indicates the bufferable, cacheable, write-through, write-back and allocate attributes of the transaction Bit-0 : Bufferable (B) Bit-1 : Cacheable (C) Bit-2 : Read Allocate (RA) Bit-3 : Write Allocate (WA) The combination where C=0 and WA/RA=1 are reserved
P7	S_AXI_AWVALID	AXI4/AXI4 LITE	I	-	Write address valid. This signal indicates that valid write address and control information are available
P8	S_AXI_AWREADY	AXI4/AXI4 LITE	O	0	Write address ready. This signal indicates that the slave is ready to accept an address and associated control signals
AXI Write Data Channel Signals					
P9	S_AXI_WDATA[C_S_AXI_DATA_WIDTH]	AXI4/AXI4 LITE	I	-	Write data
P10	S_AXI_WSTB[C_S_AXI_DATA_WIDTH/8-1:0]	AXI4/AXI4 LITE	I	-	Write strobes. This signal indicates which byte lanes in S_AXI_WDATA are/is valid
P11	S_AXI_WLAST ⁽¹⁾	AXI4	I	-	Write last. This signal indicates the last transfer in a write burst
P12	S_AXI_WVALID	AXI4/AXI4 LITE	I	-	Write valid. This signal indicates that valid write data and strobes are available
P13	S_AXI_WREADY	AXI4/AXI4 LITE	O	0	Write ready. This signal indicates that the slave can accept the write data
AXI Write Response Channel Signals					
P14	S_AXI_BID[C_S_AXI_ID_WIDTH-1:0] ⁽¹⁾	AXI4	O	0	Write response ID. This signal is the identification tag of the write response. The BID value must match the AWID value of the write transaction to which the slave is responding

Table 2: IO Signal Description (Cont'd)

Port	Signal Name	Interface	I/O	Initial State	Description
P15	S_AXI_BRESP[1:0]	AXI4/AXI4 LITE	O	0	Write response. This signal indicates the status of the write transaction "00" - OKAY "01" - EXOKAY - NA "10" - SLVERR - NA "11" - DECERR - NA
P16	S_AXI_BVALID	AXI4/AXI4 LITE	O	0	Write response valid. This signal indicates that a valid write response is available
P17	S_AXI_BREADY	AXI4/AXI4 LITE	I	-	Response ready. This signal indicates that the master can accept the response information
AXI Read Address Channel Signals					
P18	S_AXI_ARID[C_S_AXI_ID_WIDTH-1:0] ⁽¹⁾	AXI4	I	-	Read address ID. This signal is the identification tag for the read address group of signals
P19	S_AXI_ARADDR[C_S_AXI_ADDR_WIDTH-1:0]	AXI4/AXI4 LITE	I	-	Read address. The read address bus gives the initial address of a read burst transaction
P20	S_AXI_ARLEN[7:0] ⁽¹⁾	AXI4	I	-	Burst length. The burst length gives the exact number of transfers in a burst.
P21	S_AXI_ARSIZE[2:0] ⁽¹⁾	AXI4	I	-	Burst size. This signal indicates the size of each transfer in the burst.
P22	S_AXI_ARBURST[1:0] ⁽¹⁾	AXI4	I	-	Burst type. The burst type, coupled with the size information, details how the address for each transfer within the burst is calculated.
P23	S_AXI_ARVALID	AXI4/AXI4 LITE	I	-	Read address valid. This signal indicates, when HIGH, that the read address and control information is valid and will remain stable until the address acknowledgement signal, ARREDY, is high.
P24	S_AXI_ARREADY	AXI4/AXI4 LITE	O	0	Read address ready. This signal indicates that the slave is ready to accept an address and associated control signals.
AXI Read Data Channel Signals					
P25	S_AXI_RID[C_S_AXI_ID_WIDTH-1:0] ⁽¹⁾	AXI4	O	0	Read ID tag. This signal is the ID tag of the read data group of signals. The RID value is generated by the slave and must match the ARID value of the read transaction to which it is responding.
P26	S_AXI_RDATA[C_S_AXI_DATA_WIDTH-1:0]	AXI4/AXI4 LITE	O	0	Read data
P27	S_AXI_RRESP[1:0]	AXI4/AXI4 LITE	O	0	Read response. This signal indicates the status of the read transfer.
P28	S_AXI_RLAST ⁽¹⁾	AXI4	O	0	Read last. This signal indicates the last transfer in a read burst
P29	S_AXI_RVALID	AXI4/AXI4 LITE	O	0	Read valid. This signal indicates that the required read data is available and the read transfer can complete
P30	S_AXI_RREADY	AXI4/AXI4 LITE	I	-	Read ready. This signal indicates that the master can accept the read data and response information
AXI Lite Register Interface⁽²⁾					

Table 2: IO Signal Description (Cont'd)

Port	Signal Name	Interface	I/O	Initial State	Description
AXI Write Address Channel Signals⁽²⁾					
P31	S_AXI_CTRL_AWADDR[C_S_AXI_CTRL_ADDR_WIDTH-1:0] ⁽²⁾	AXI4LITE	I	-	AXI Write address for register interface. The write address bus gives the address of the write transaction
P32	S_AXI_CTRL_AWVALID ⁽²⁾	AXI4LITE	I	-	Write address valid for register interface. This signal indicates that valid write address and control information are available
P33	S_AXI_CTRL_AWREADY ⁽²⁾	AXI4LITE	O	0x0	Write address ready for register interface. This signal indicates that the slave is ready to accept an address and associated control signals
AXI Write Data Channel Signals⁽²⁾					
P34	S_AXI_CTRL_WDATA[C_S_AXI_CTRL_DATA_WIDTH - 1: 0] ⁽²⁾	AXI4LITE	I	-	Write data for register interface
P35	S_AXI_CTRL_WSTB[C_S_AXI_CTRL_DATA_WIDTH/8-1:0] ⁽²⁾	AXI4LITE	I	-	Write strobes for register interface. This signal indicates which byte lanes to update in memory
P36	S_AXI_CTRL_WVALID ⁽²⁾	AXI4LITE	I	-	Write valid for register interface. This signal indicates that valid write data and strobes are available
P37	S_AXI_CTRL_WREADY ⁽²⁾	AXI4LITE	O	0x0	Write ready for register interface. This signal indicates that the slave can accept the write data
AXI Write Response Channel Signals⁽²⁾					
P38	S_AXI_CTRL_BRESP[1:0] ⁽²⁾	AXI4LITE	O	0x0	Write response for register interface. This signal indicates the status of the write transaction "00" - OKAY "10" - SLVERR
P39	S_AXI_CTRL_BVALID ⁽²⁾	AXI4LITE	O	0x0	Write response valid for register interface. This signal indicates that a valid write response is available
P40	S_AXI_CTRL_BREADY ⁽²⁾	AXI4LITE	I	-	Response ready for register interface. This signal indicates that the master can accept the response information
AXI Read Address Channel Signals⁽²⁾					
P41	S_AXI_CTRL_ARADDR[C_S_AXI_CTRL_ADDR_WIDTH - 1: 0] ⁽²⁾	AXI4LITE	I	-	Read address for register interface. The read address bus gives the address of a read transaction
P42	S_AXI_CTRL_ARVALID ⁽²⁾	AXI4LITE	I	-	Read address valid for register interface. This signal indicates, when HIGH, that the read address and control information is valid and will remain stable until the address acknowledgement signal, ARREDY, is high.
P43	S_AXI_CTRL_ARREADY ⁽²⁾	AXI4LITE	O	0x1	Read address ready for register interface. This signal indicates that the slave is ready to accept an address and associated control signals.
AXI Read Data Channel Signals⁽²⁾					
P44	S_AXI_CTRL_RDATA[C_S_AXI_CTRL_DATA_WIDTH - 1: 0] ⁽²⁾	AXI4LITE	O	0x0	Read data for register interface

Table 2: IO Signal Description (Cont'd)

Port	Signal Name	Interface	I/O	Initial State	Description
P45	S_AXI_CTRL_RRESP[1:0] ⁽²⁾	AXI4LITE	O	0x0	Read response for register interface. This signal indicates the status of the read transfer. "00" - OKAY "10" - SLVERR
P46	S_AXI_CTRL_RVALID ⁽²⁾	AXI4LITE	O	0x0	Read valid for register interface. This signal indicates that the required read data is available and the read transfer can complete
P47	S_AXI_CTRL_RREADY ⁽²⁾	AXI4LITE	I	-	Read ready for register interface. This signal indicates that the master can accept the read data and response information
System Ports					
P48	MPLB_Clk	System	I	-	PLB clock to the secondary side of the bridge
P49	MPLB_Rst	System	I	-	PLB reset
P50	Bridge_Interrupt	System	O	0	Error interrupt for bufferable AXI write transactions
PLB Master I/O Signals					
P51	M_request	PLB	O	0	Bus request the arbiter
P52	M_RNW	PLB	O	0	PLB read not write
P53	M_BE[0:(C_MPLB_DWIDTH - 1/8) - 1]	PLB	O	0	Master byte enables
P54	M_Msize[0:1]	PLB	O	0	Master data bus size "00" - 32-bit Master (if C_MPLB_NATIVE_DWIDTH=32) "01" - 64 bit Master (if C_MPLB_NATIVE_DWIDTH=64)
P55	M_size[0:3]	PLB	O	0	Master transfer size "0000" - Singles - M_BE determines byte line. Always "0000" if C_AXI_TYPE=0 "0001" - 4 word Cacheline - M_BE ignored "0010" - 8 word Cacheline - M_BE ignored Fixed length burst of data width that do not exceed either the values of C_MPLB_NATIVE_DWIDTH or C_MPLB_DWIDTH Burst transfer - length determined by M_BE "1000" - Byte burst - Not supported "1001" - Half word burst - Not supported "1010" - Word burst "1011" - Double word burst - Supported if Slave native data width is 64-bit "1100" - Quad word burst - Not supported "1100" - Octal word burst - Not supported
P56	M_type[0:2]	PLB	O	0	Master transfer type Driven to logic low ²⁾ "000" - Memory transfer (only supported)
P57	M_ABus[0:(C_MPLB_AWIDTH - 1)]	PLB	O	0	Master address bus
P58	M_wrBurst	PLB	O	0	Master burst write transfer indicator
P59	M_rdBurst	PLB	O	0	Master read write transfer indicator

Table 2: IO Signal Description (Cont'd)

Port	Signal Name	Interface	I/O	Initial State	Description
P60	M_WrDBus[0:(C_MPLB_DWIDTH - 1)]	PLB	O	0	Master write data bus
P61	PLB_MaddrAck	PLB	I	-	PLB master address acknowledge
P62	PLB_MSSize[0:1]	PLB	I	-	PLB slave data bus size
P63	PLB_MTimeout	PLB	I	-	PLB master bus time out
P64	PLB_MRdErr	PLB		-	PLB master slave read error indicator
P65	PLB_MWrErr	PLB	I	-	PLB master slave write error indicator
P66	PLB_MRdDBus[0:(C_MPLB_DWIDTH - 1)]	PLB	I	-	PLB master read data bus
P67	PLB_MRdDAck	PLB	I	-	PLB master read data acknowledge
P68	PLB_RdBTerm	PLB	I	-	PLB master terminate read burst indicator
P69	PLB_MWrDAck	PLB	I	-	PLB master write data acknowledge
PLB Master Unused Output Signals - driven default					
P70	M_TAttribute[0:15] ⁽³⁾	PLB	O	0	Unused PLB master transfer attributes
P71	M_lockErr ⁽³⁾	PLB	O	0	Unused PLB master lock error
P73	M_abort ⁽³⁾	PLB	O	0	Unused PLB master abort
P73	M_UABus[0:(C_MPLB_AWIDTH - 1)] ⁽³⁾	PLB	O	0	Unused PLB master upper bits of address bus
P74	MD_Error ⁽³⁾	PLB	O	0	Master error detection indicator Unsupported feature; port driven to logic low ²⁾
P75	M_priority[0:1] ⁽³⁾	PLB	O	0	Bus request priority Driven to logic low ²⁾ M_priority will be assigned the binary conversion of C_PLB_MPRIORITY "11" - Highest priority "10" - Next highest "01" - Next highest "00" - Lowest
P76	M_buslock ⁽³⁾	PLB	O	0	Bus lock request
PLB Master Unused Input Signals					
P77	PLB_MRdWdAddr[0:3]	PLB	I	-	PLB master read word address
P78	PLB_MBusy	PLB	I	-	Unused PLB master slave busy indicator
P79	PLB_MIRQ	PLB	I	-	Unused

Table 2: IO Signal Description (Cont'd)

Port	Signal Name	Interface	I/O	Initial State	Description
P80	PLB_Mrearbitrate	PLB	I	-	PLB master bus rearbitrate indicator
P81	PLB_MWrbTerm	PLB	I	-	PLB master terminate write burst indicator

Notes:

- Valid only when C_S_AXI_PROTOCOL="AXI4"
- Valid only when C_S_AXI_PROTOCOL="AXI4" AND C_S_AXI_SUPPORTS_WRITE=1 and C_EN_DEBUG_REG=1
- Unused port. Output will have default assignment.

Allowable Parameter Combinations

The current implementation of the PLBv46 Master Burst has the following restrictions that apply to parameter value settings:

- The assigned value for C_MPLB_NATIVE_DWIDTH should be same as C_S_AXI_DATA_WIDTH.

Parameter - I/O Signal Dependencies

The dependencies between the AXI to PLBv46 Bridge core design parameters and I/O signals are described in Table 3.

Table 3: Parameter - I/O Signal Dependencies

Generic or Port	Name	Affects	Depends	Relationship Description
Design Parameters				
G3	C_EN_DEBUG_REG	G17-G20, P31-P47	G4, G10	C_EN_DEBUG_REG is invalid when C_S_AXI_PROTOCOL="AXI4LITE" or C_S_AXI_SUPPORTS_WRITE=0. G17-G20 & P31-P47- invalid when C_EN_DEBUG_REG=0
G4	C_S_AXI_PROTOCOL	G3, G5-G7, G10-G14, P1, P3-P6, P11, P14, P18, P20, P25, P28, P58, P59	-	All are invalid if C_S_AXI_PROTOCOL="AXI4LITE" Input ports are unused output ports are driven to their default
G5	C_S_AXI_SUPPORTS_NARROW_BURST	P4, P24	G4	C_S_AXI_SUPPORTS_NARROW_BURST is invalid when C_S_AXI_PROTOCOL="AXI4LITE" P4 & P24 considered having constant value corresponding to AXI data width when C_S_AXI_SUPPORTS_NARROW_BURST=0
G6	C_S_AXI_SUPPORTS_EXCL_ACCESS	-	G4	Invalid when C_S_AXI_PROTOCOL="AXI4LITE"
G7	C_S_AXI_ID_WIDTH	P3, P14, P18, P25,	G4	C_S_AXI_ID_WIDTH is invalid when C_S_AXI_PROTOCOL="AXI4LITE" Defines the width of the ports
G8	C_S_AXI_ADDR_WIDTH	P2, P19	-	Defines the width of the ports
G9	C_S_AXI_DATA_WIDTH	P9, P10, P44	-	Defines the width of the ports

Table 3: Parameter - I/O Signal Dependencies (Cont'd)

Generic or Port	Name	Affects	Depends	Relationship Description
G10	C_S_AXI_SUPPORTS_WRITE	P1-P17, P58	G4	C_S_AXI_SUPPORTS_WRITE is invalid if C_S_AXI_PROTOCOL="AXI4LITE" P1-P17, P58 are invalid if C_S_AXI_SUPPORTS_WRITE=0 Input ports are unused output ports are driven to their default
G11	C_S_AXI_SUPPORTS_READ	P18-P30, P59	G4	C_S_AXI_SUPPORTS_READ is invalid if C_S_AXI_PROTOCOL="AXI4LITE" P18-P30, P59 are invalid if C_S_AXI_SUPPORTS_READ=0 Input ports are unused output ports are driven to their default
G12	C_S_AXI_WRITE_ACCEPTANCE	-	G8	Invalid if C_S_AXI_PROTOCOL="AXI4LITE"
G13	C_S_AXI_READ_ACCEPTANCE	-	G8	Invalid if C_S_AXI_PROTOCOL="AXI4LITE"
G14	C_S_AXI_SUPPORTS_BARRIERS	-	G8	Invalid if C_S_AXI_PROTOCOL="AXI4LITE"
G17	C_S_AXI_CTRL_ADDR_WIDTH	P31, P41	G2, G8, G10	Invalid if C_S_AXI_PROTOCOL="AXI4LITE" or C_EN_DEBUG_REG=0 or C_S_AXI_SUPPORTS_WRITE=0 Port width depends on the generic
G18	C_S_AXI_CTRL_DATA_WIDTH	P34, P35, P44	G2, G8, G10	Invalid if C_S_AXI_PROTOCOL="AXI4LITE" or C_EN_DEBUG_REG=0 or C_S_AXI_SUPPORTS_WRITE=0 Port width depends on the generic
G10	C_S_AXI_CTRL_BASEADDR	-	G2, G8, G10	Invalid if C_S_AXI_PROTOCOL="AXI4LITE" or C_EN_DEBUG_REG=0 or C_S_AXI_SUPPORTS_WRITE=0
G20	C_S_AXI_CTRL_HIGHADDR	-	G2, G8, G10	Invalid if C_S_AXI_PROTOCOL="AXI4LITE" or C_EN_DEBUG_REG=0 or C_S_AXI_SUPPORTS_WRITE=0
G21	C_MPLB_AWIDTH	P57, P73	-	Port width depends on the generic
G22	C_MPLB_DWIDTH	P53, P60, P66	-	Port width depends on the generic
G23	C_MPLB_NATIVE_DWIDTH	-	G9	Must be same as C_S_AXI_DATA_WIDTH

Design Details

Bridge Transaction Translation

The PLB supported AXI transactions are directly translated to PLB. For some translations, multiple PLB transactions must be performed. For instance, PLB does not allow burst length more than 16, but AXI allows up to 256. Deasserted byte enables (BEs) during burst transfer is not allowed for PLB, but AXI does allow this. The AXI to PLB transactions translation is shown in [Table 4](#).

Table 4: AXI to PLB Transaction Translation

AXI Transaction	PLB Transaction	Note
Write: Burst 1 Word, Half Word, Byte	Single Write	The byte address bits will be set based on the first byte enable that is asserted, as required by PLB protocol.
Read: Burst 1 Word, Half Word, Byte	Single Read	The byte address bits will be aligned to the word boundary.
INCR/FIXED Write: Burst 2-16 word transfers	This can break into write transaction (max3) as below: 1. Single, Single 2. Single, Single, Single 3. Single, Burst, Single 4. Single, Burst 5. Burst, Single 6. Burst	If all write strobes are not asserted on the first and/or last word, then PLB single write will be performed on the first and/or last word. The byte address bits of the first word single PLB transaction will be set based on the first byte enable that is asserted, as required by PLB protocol. Address incrementing will be performed as necessary to the single and burst transaction: 1. From Single to next transfer (single or burst) address is incremented by 0x04 and aligned to word boundary. 2. From Burst to next transfer (single) address is incremented by length of transfer during burst i.e. (M_BE + '1').
INCR/FIXED Read: burst 2-16 word transfers	Burst read 2-16 word transfers	The start address is aligned to the word boundary.
WRAP: 2 word write	Burst: 2 word write/read	Data reordering to start from address align to cache during write and target word first during read is performed in the bridge.
WRAP: 4, 8 word write	4, 8 word burst write	
WRAP: 16 word write	Two 8 word burst write	Data reordering to start from address align to cache during write and target word first during read is performed in the bridge. The first transfer will be from starting address with M_ABus(5) = '0'. The second transfer will be from starting address with M_ABus(5) = '1'.
WRAP : 2,4,8,16 beat Read	Aligned Wrap - 2,4,8,16 burst read Un-aligned Wrap - Single, Burst Burst, Burst Burst, Single	The read on the PLB is always generated from the starting AXI wrap address. If wrap transfer is not starting from the Wrap bounding, the core breaks the burst transaction on the wrap boundary.

Table 4: AXI to PLB Transaction Translation

AXI Transaction	PLB Transaction	Note
INCR Write: burst 2 half word transfers	The possible PLB transfers: 1. Single, Single 2. Single	This will be converted to word transfer by aggregating the half words. On PLB one single transfer is performed if S_AXI_AWADDR(1)='0' else two singles. If all the valid write strobes are not asserted on the first and/or last word, then PLB single write will be performed on the first and/or last word. The byte address bits of the first word single PLB transaction will be set based on the first byte enable that is asserted, as required by PLB protocol. Address incrementing will be performed as necessary to the single transactions: From Single to second single address is incremented by 0x04 and aligned to word boundary.
INCR Write: burst 3 half word transfers	This will break into two singles on PLB	This will be converted to word transfer by aggregating the half words. The byte address bits of the first word single PLB transaction will be set based on the first byte enable that is asserted, as required by PLB protocol. Address incrementing will be performed as necessary to the single and burst transaction: From Single to second single address is incremented by 0x04 and aligned to word boundary.
INCR Write: burst 4-16 half word transfers	The max burst length of this would be 9 as half words will be converted into words. This can break into write transaction (max3) as below: 1. Single, Single 2. Single, Single, Single 3. Single, Burst, Single 4. Single, Burst 5. Burst, Single 6. Burst	This will be converted to word transfer by aggregating the half words. If all valid write strobes are not asserted on the first and/or last word, then PLB single write will be performed on the first and/or last word. The byte address bits of the first word single PLB transaction will be set based on the first byte enable that is asserted, as required by PLB protocol. Address incrementing will be performed as necessary to the single and burst transaction: 1. From Single to next transfer (single or burst) address is incremented by 0x04 and aligned to word boundary. 2. From Burst to next transfer (single) address is incremented by length of transfer during burst i.e. (M_BE + '1').
INCR Read: burst 2-16 Half Word transfers	Burst read 2-9 word	This will be converted to word transfer and S_AXI_RDATA will have same value for two S_AXI_RREADY cycles. The max burst length of this would be 9 as half words will be converted into words.
INCR Write: burst 2-4 byte transfers	The possible PLB transfers: 1. Single, Single 2. Single	This will be converted to word transfer by aggregating the bytes. On PLB one single transfer is performed if all the bytes fall in the same word else two singles. The byte address bits of the first word single PLB transaction will be set based on the first byte enable that is asserted, as required by PLB protocol. From Single to second single address is incremented by 0x04 and aligned to word boundary.

Table 4: AXI to PLB Transaction Translation

AXI Transaction	PLB Transaction	Note
INCR Write: burst 5-16 bytes transfers	<p>The max burst length of this would be 5 as bytes will be converted into words.</p> <p>This can break into write transaction (max3) as below:</p> <ol style="list-style-type: none"> 1. Single, Single 2. Single, Single, Single 3. Single, Burst, Single 4. Single, Burst 5. Burst, Single 6. Burst 	<p>This will be converted to word transfer by aggregating the bytes. The byte address bits of the first word single PLB transaction will be set based on the first byte enable that is asserted, as required by PLB protocol.</p> <p>Address incrementing will be performed as necessary to the single and burst transaction:</p> <ol style="list-style-type: none"> 1. From Single to next transfer (single or burst) address is incremented by 0x04 and aligned to word boundary. 2. From Burst to next transfer (single) address is incremented by length of transfer during burst i.e. (M_BE + '1').
INCR Read: burst 2-16 Bytes transfers	Burst read 2-5 word	<p>This will be converted to word transfer and S_AXI_RDATA will have same value for four S_AXI_RREADY cycles.</p> <p>The max burst length of this would be 5 as bytes will be converted into words.</p>
FIXED: Write/Read burst 2-16 - Half Word/Byte	Singles - Write/Read	Number of singles requested on PLB is equal to the burst length requested by AXI
WRAP: Write/Read	Singles - Write/Read	Data reordering to start from address align to cache during write and target word first during read is performed in the bridge.
<ol style="list-style-type: none"> 1. burst 2 - Half word 2. burst 2/4 - bytes 		
WRAP: Write/Read	Burst - 2 Write/Read	All wrap transfers will be terminated in PLB as burst transfers.
<ol style="list-style-type: none"> 1. burst 4 - Half word 2. burst 8 - bytes 		
WRAP: Write	4 Word Burst - Write	
<ol style="list-style-type: none"> 1. burst 8 - Half word 2. burst 16 - bytes 		
WRAP: Write - burst 16 - Half word	8 Word Burst - Write/Read	
WRAP : 2,4,8,16 beat Read	<p>Aligned Burst - 2,4,8,16 burst read</p> <p>Un-aligned Wrap - Single , Burst</p> <p>Burst, Burst</p> <p>Burst, Single</p>	All wrap transfers will be terminated in PLB as burst transfers. The read on the PLB is always generated from the starting AXI wrap address. If wrap transfer is not starting from the Wrap bounding, the core breaks the burst transaction on the wrap boundary.

Notes:

1. In AXI - INCR/FIXED write transactions, deasserted write strobes are supported only in the first and last word of the burst write.
2. All valid write strobes must to HIGH for a write WRAP transfer.

Table 5: AXI to PLBv46 Transaction Translation (C_S_AXI_DATA_WIDTH= 64 and C_PLB_SMALLEST_SLAVE_SIZE = 32)

AXI Transaction	PLB Transaction	Note
INCR/FIXED Write: Burst 1 double word transfers	Bridge will generate conversion cycle on PLB This can break into write transaction (max2) as below: 1. Single 2. Single, Single	The byte address bits will be set based on the first byte enable that is asserted, as required by PLB protocol.
INCR/FIXED Write: Burst 2-3 double word transfers	Bridge will generate conversion cycle on PLB and will adjust the burst length dynamically This can break into write transaction (max4) as below: 1. Single, Single 2. Single, Single, Single 2. Single, Single, Single, Single 3. Single, Single, Burst 4. Burst, Single, Single 5. Single, Single, Single, Single, Single, Single 6. Burst	If all write strobes are not asserted on the first and/or last word, then PLB singles write will be performed on the first and/or last double word. The byte address bits of the first word single PLB transaction will be set based on the first byte enable that is asserted, as required by PLB protocol. Address incrementing will be performed as necessary to the single and burst transaction: 1. From Single to next transfer (single or burst) address is incremented by 0x04 and aligned to word boundary. 2. From Burst to next transfer (single) address is incremented by length of transfer during burst i.e. (M_BE + '1').
INCR/FIXED Write: Burst 4-16 double word transfers	Bridge will generate conversion cycle on PLB and will adjust the burst length dynamically This can break into write transaction (max5) as below: 1. Single, Single, Burst, Single, Single 2. Single, Single, Burst 3. Burst, Single, Single 4. Burst	If all write strobes are not asserted on the first and/or last word, then PLB singles write will be performed on the first and/or last double word. The byte address bits of the first word single PLB transaction will be set based on the first byte enable that is asserted, as required by PLB protocol. Address incrementing will be performed as necessary to the single and burst transaction: 1. From Single to next transfer (single or burst) address is incremented by 0x04 and aligned to word boundary. 2. From Burst to next transfer (single) address is incremented by length of transfer during burst i.e. (M_BE + '1').
INCR Write: Burst 17-256 double word transfers	Bridge will generate conversion cycle on PLB and will adjust the burst length dynamically This can break into write transaction (max5) as below: 1. Single, Single, Burst(n), Single, Single 2. Single, Single, Burst(n) 3. Burst(n), Single, Single	If all write strobes are not asserted on the first and/or last word, then PLB singles write will be performed on the first and/or last double word. The byte address bits of the first word single PLB transaction will be set based on the first byte enable that is asserted, as required by PLB protocol. Address incrementing will be performed as necessary to the single and burst transaction: 1. From Single to next transfer (single or burst) address is incremented by 0x04 and aligned to word boundary. 2. From Burst to next transfer (single) address is incremented by length of transfer during burst i.e. (M_BE + '1').
Read: Burst 1 double word transfer	Bridge will generate conversion cycle on PLB This can break into write transaction (max2) as below: 1. Single 2. Single, Single	The byte address bits will be aligned to the word boundary.
INCR/FIXED Read: Burst 2-8 double word transfers	Burst read 4-16 word transfers	The start address is aligned to the word boundary.

Table 5: AXI to PLBv46 Transaction Translation (C_S_AXI_DATA_WIDTH= 64 and C_PLB_SMALLEST_SLAVE_SIZE = 32)

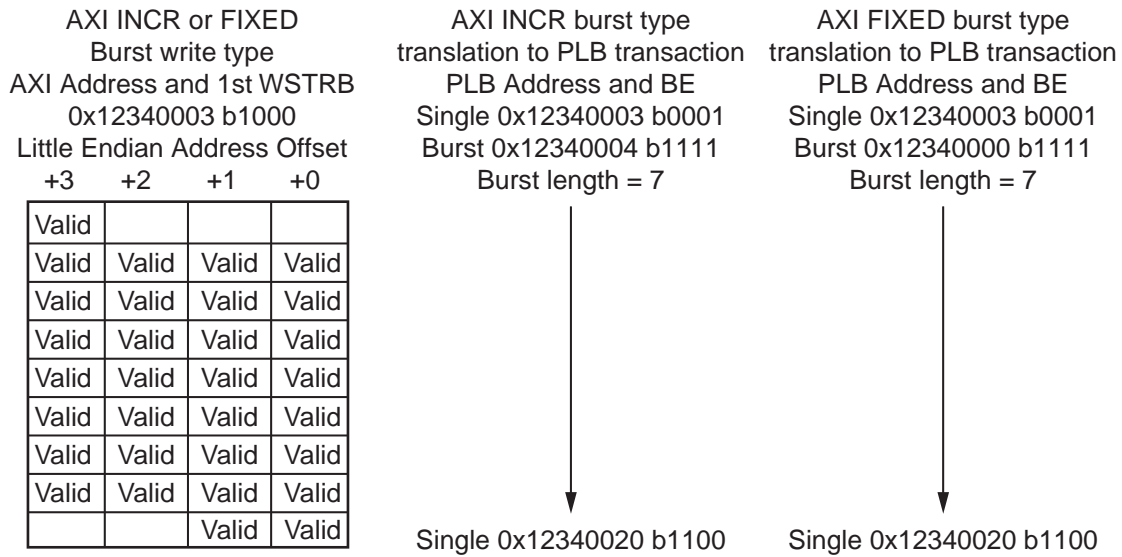
AXI Transaction	PLB Transaction	Note
INCR/FIXED Read: burst 9-16 double word transfers	Burst read 16 word + Burst read of 2-16 word transfers	The start address is aligned to the word boundary.
INCR Read: burst 17-256 double word transfers	Burst read (17-256)*2/16 transaction + Burst read of 2-16 word transfers	The start address is aligned to the word boundary.
WRAP: 2 double word write/read	Burst: 2 word write/read	Data reordering to start from address align to cache during write and target word first during read is performed in the bridge.
WRAP: 4, 8 double word write/read	4, 8 word burst write/read	
WRAP: 16 double word write/read	Two 8 word burst write/read	All wrap transfers will be terminated in PLB as burst transfers. Data reordering to start from address align to cache during write and target word first during read is performed in the bridge. The first transfer will be from starting address with M_ABus(5) = '0'. The second transfer will be from starting address with M_ABus(5) = '1'.

AXI Write Burst Without All Write Strobes Asserted

AXI allows write strobes to be deasserted on write bursts for any data transfer. An optimization to the AXI to PLB bridge is that it is designed to handle write strobes not all (valid bits) asserted in a given transaction only in the first and last word of the write burst of type FIXED and INCR on AXI. The bridge assumes that the remaining (other than first and last data beat in a transaction) will be always HIGH. If either or both first and last word transfers do not have all write strobes asserted, then the PLB single transactions will be performed for the first and/or last word to allow the BE information to be passed to the PLB slave. [Figure 3](#) shows how burst writes with write strobes not all asserted for the first and last word are translated to the PLB-side for both INCR and FIXED type AXI burst write transactions.

AXI to PLBv46 Bridge is not validating the intermediate write strobes (only first and last are considered) during burst, the bridge will not generate any byte mask on PLBv46 slave interface for the address in between the burst; this will overwrite the complete 32/64-bit data on a given address irrespective how the data strobes are generated by AXI Master.

For AXI WRAP transfer all the valid (depend on S_AXI_AWSIZE) write strobes must be HIGH.



DS712_03

Figure 3: AXI (32-bit) write burst type of INCR and FIXED translation to PLB (32-bit) transactions

AXI Narrow Transactions

For the transaction where size is narrower than data width are treated as narrow transfers. If the narrow transfer is "FIXED" burst type, then the byte address will remain constant for all singles required to complete the entire burst. If the narrow transfer is "INCR" type, then the narrow data received from AXI will be collapsed to create a complete data beat (if possible). For detail transaction translation from AXI to PLB refer [Table 4](#).

AXI WRAP Transactions

An optimization to the AXI to PLBv46 Bridge, all the WRAP transfers from AXI are converted as single or burst transfer on PLBv46.

AXI allows for the target word to be any word address in the WRAP transfer. The AXI to PLBv46 Bridge performs re-ordering of AXI target word write data to PLB line word first write data.

For AXI WRAP reads that are not line word first - AXI to PLBv46 Bridge can generate two read request on PLB.

For detail translation from AXI WRAP to PLB single/burst refer [Table 4](#).

Address Pipelining

The parameters C_S_AXI_WRITE_ACCEPTANCE and C_S_AXI_READ_ACCEPTANCE define the number of address and control information that can be buffered (max 2) for each read and write. When the ACCEPTANCE parameter is set to 1, the next address will not be accepted until the response phase and the transfer on PLB of the first is completed. When the ACCEPTANCE parameter is set to 2, the next address will be accepted irrespective of the transfer complete of first. But the third address will be accepted only if at least the response phase and the transfer on PLB of the first transaction is completed.

The AXI to PLBv46 Bridge does not support the pipelined address on PLB. This means the slaves that are accessed by the AXI to PLBv46 Bridge should not respond to secondary request (SAVALID) both for read and write.

AXI - INCR/WRAP, Narrow Transfers

AXI support incremental burst transfer of bytes and half word. This transfers are not supported as per Xilinx PLB v4.6 simplification.

To optimize and obtain better throughput :

For Writes - All the AXI data will be collapsed to convert to data beat equal to size of data bus (wherever possible) and a burst will be initiated on PLB

For Reads - A burst will be initiated on PLB and the S_AXI_RVALID will be asserted for more than one cycle keeping same data on S_AXI_RDATA. For example for the INCR, byte burst from AXI of length four that start with address aligned to word boundary (e.g 0x0, 0x4, 0x8, 0xC) the S_AXI_RDATA will have same value for four cycles of S_AXI_RREADY assertion.

Endian Support

The endian conversion is implemented in the design depending on C_EN_BYTE_SWAP.

If C_EN_BYTE_SWAP=0. The possible connection from AXI to PLB for 32-bit data width is as below.

- AXI is little endian and PLB is big endian
 - M_ABUS(0 to 31) = S_AXI_AxADDR(31 down to 0)
 - M_WrDBUS(0 to 31) = S_AXI_AWDATA(31 down to 0)
 - M_BE(0 to 3) = S_AXI_WSTB(3 down to 0)
 - S_AXI_AWDATA(31 down to 0) = PLB_MRdDBUS(0 to 31)
- AXI is big endian and PLB is big endian
 - M_ABUS(0 to 31) = S_AXI_AxADDR(0 to 31)
 - M_WrDBUS(0 to 31) = S_AXI_AWDATA(0 to 31)
 - M_BE(0 to 3) = S_AXI_WSTB(0 to 3)
 - S_AXI_AWDATA(0 to 31) = PLB_MRdDBUS(0 to 31)

Byte Invariance

Byte invariance is implemented if C_EN_BYTE_SWAP=1.

AXI is little endian and PLB is big endian. The AXI to PLBv46 Bridge maintains byte invariance, or using Xilinx IP terminology, byte addressing integrity is maintained in the bridge design. This means that 32-bit word data from any address on the PLBv46 bus has the bytes swapped in traversing the bridge so that byte data of byte lanes of the same numerical address offsets yields the same byte data when read by the little endian AXI-side or by a remote master on the big endian PLB-side. For byte transactions, any byte addressed data read from the AXI side or the PLB side yields the same byte of data. Write strobe signals from the AXI master port are similarly swapped. Byte and strobe swapping are shown in [Figure 4](#).

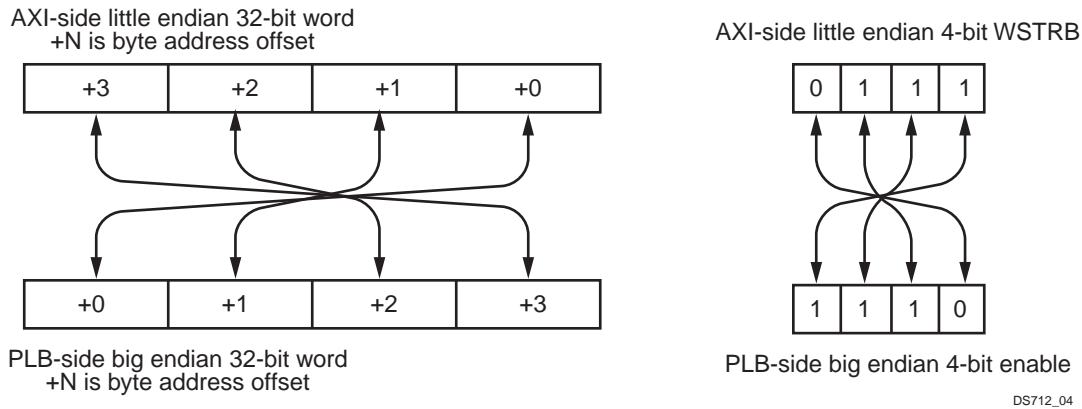


Figure 4: Byte DataSwap and WrSTRB Swap to BEs as Data Traverses AXI to PLBv46 Bridge

Read and Write Interaction

Consecutive Read and Write transactions or vice versa to the same address issued by AXI, are directly transferred to PLB, as PLB does not support out of order transactions.

AXI Trustzone and Protection Unit Support

The AXI to PLBv46 Bridge does not support Trustzone. As a consequence, The AR(W)PROT input to the AXI slave port will be ignored and all requests will be responded to. If master port that the AXI to PLBv46 Bridge is connected to is configured as a secure port and a master attempts a non-secure transaction to the AXI to PLBv46 Bridge, then the interconnect will not present the transaction to the bridge. As a result, the transaction will not be presented on the PLB bus.

AXI signals AR(W)PROT[0] and [2] will have no effect on AXI to PLB behavior and the resulting PLB transaction. Bit 0 indicates normal or privileged access, but the PLB does not have any such qualifiers, hence, the response is the same for normal or privileged accesses. Bit 2 indicates data or instruction access, which again, the PLB does not qualify and the bridge response is the same for both data and instruction accesses.

AXI Atomic Accesses

The AXI to PLBv46 Bridge does not support AXI atomic exclusive accesses.

PLBv46 Error Conditions - Read and Non-bufferable Write transactions

The bridge executes posted writes and both write and read addresses are pipelined/buffered in the bridge. The write response (for a non-bufferable transaction) to the AXI will be generated after all the data are received by PLB or a timeout is generated by PLB. The read response will be sent along with the data as per AXI protocol.

- Slave Error - PLB_Wr_Err/PLB_Rd_Err from PLB causes the ERROR response to AXI.
- Decode Error - Address phase timeout (assertion of PLB_MTimeout) will cause DECERR response to AXI. During read along with the response S_AXI_RVALID and S_AXI_RLAST are asserted as per AXI protocol.

PLBv46 Error Conditions - Bufferable Write Transfer

The bridge executes posted writes and generates an early response (after the assertion of S_AXI_WLAST) to AXI for the cacheable transactions. There is a possibility of having an error or timeout condition on PLB for this transaction. But since the response is sent early now there is no mechanism to inform AXI master about the failure.

To overcome this situation registers are implemented in the design that captures the address and other control information of such errored transaction and will generate an interrupt.

The more detail about these registers and interrupt is detailed in the sub-sections below.

Register Descriptions

Table 6 shows all the AXI to PLBv46 Bridge registers and their addresses. All the registers described in sections below are implemented only when C_S_AXI_PROTOCOL="AXI4" AND C_S_AXI_SUPPORTS_WRITE=1 AND C_EN_DEBUG_REG=1.

Table 6: AXI to PLBv46 Bridge Registers ⁽¹⁾

Base Address + Offset (hex)	Register Name	Access Type	Default Value (hex)	Description
C_S_AXI_CTRL_BASEADDR + 0x0	BESR	R ⁽²⁾	0x0	Bridge Error Status Register
C_S_AXI_CTRL_BASEADDR + 0x4	BEAR	R ⁽³⁾	0x0	Bridge Error Address Register
C_S_AXI_CTRL_BASEADDR + 0x8	DGIE	R/W	0x0	Device Global Interrupt Enable Register
C_S_AXI_CTRL_BASEADDR + 0xC	DIER	R/W	0x0	Device Interrupt Enable Register

Notes:

1. The registers are included only when C_EN_DEBUG_REG is set to 1.
2. This register is cleared after read access to this register.
3. Read only register. Writing into this register has no effect.

Bridge Error Status Register (BESR) and Bridge Error Address Register (BEAR)

The following section details the register descriptions of the BESR and BEAR. These registers are included only when C_EN_DEBUG_REG = 1.

They are used to provide transaction error information to the user application, typically software. When these registers are enabled, PLB_Wr_Err or PLB_MTimeout (timeout for write) will cause a capture trigger to occur for the BESR and the BEAR. The BESR captures the AXI transaction qualifiers and the BEAR captures the AXI address for the first offending command. Once captured, the data is retained until the user application reads the data from the registers. The BESR register gets cleared after reading.

It should be noted by the user that the, slave error or decode error can be used to generate an interrupt to the user application. This requires enabling Device Global Interrupt Enable Register and Device Interrupt Enable Register. This interrupt may then be used by the user application to signal the need to service the BESR and BEAR.

When C_EN_DEBUG_REG = 0, the strobe error and errors on PLB can't be reported to AXI. It is assumed that the user application does not issue transactions that generates errors on PLB.

The BESR is shown in Figure 5 and detailed in Table 7. The BEAR is shown in Figure 6 and detailed in Table 8.

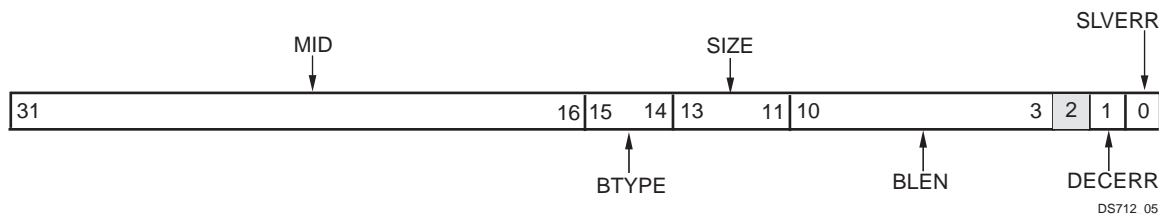


Figure 5: Bridge Error Status Register

Table 7: Bridge Error Status Register (BESR) Description ⁽¹⁾

Bit(s)	Name	Core Access	Reset Value	Description
31-16 ⁽²⁾	MID	R/W	"00"	AXI Write Transaction ID This value reflects the S_AXI_AWID qualifier at the time of error capture.
15-14	BTYPE	R/W	"00"	AXI Write Burst Type This value reflects the S_AXI_AWBURST qualifier at the time of error capture.
13-11	BSIZE	R/W	"000"	AXI Write Burst Size This value reflects the S_AXI_AWSIZE qualifier at the time of error capture.
10-3	BLLEN	R/W	"00000000"	AXI Write Burst Length This value reflect the S_AXI_AWLEN qualifier at the time of error capture.
2	Reserved	R/W	'0'	Reserved
1	DECERR	R/W	'0'	Decode Error This bit will be asserted when PLB_MTimeOut is asserted by the PLB. This indicates that there is no slave at the transaction address. '0' = No Decode Error asserted. '1' = Decode Error asserted.
0	SLVERR	R/W	'0'	Slave Error This bit will be asserted when PLB_MWrErr is asserted by the PLB. This indicates that the access has reached the PLB slave successfully, but the slave wishes to return an error condition. '0' = No Slave Error asserted. '1' = Slave Error asserted.

Notes:

1. This register is cleared after reading.
2. Vector length of MID is defined by parameter C_S_AXI_ID_WIDTH

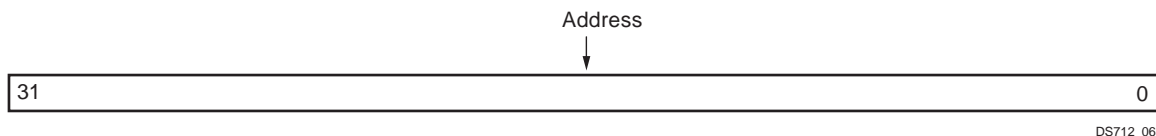


Figure 6: Bridge Error Address Register

Table 8: Bridge Error Address Register (BEAR) Description

Bit(s)	Name	Core Access	Reset Value	Description
31-0	Address(0 to 31)	R	Zeros	Transaction Address(0-31) This value reflects the S_AXI_AWADDR at the time of error capture.

Device Global Interrupt Enable Register (DGIE)

The Device Global Interrupt Enable Register provides the final enable/disable for the interrupt output and resides in the Register and Interrupt Module. It is a read/write register addressed at an offset 0x08 from base address C_S_AXI_CTRL_BASEADDR. If interrupts are globally disabled (the DGIE bit is set to '0', there will be no interrupt from the device under any circumstances. This is a single bit read/write register as shown in Figure 7. The DGIE bit definitions is shown in Table 9

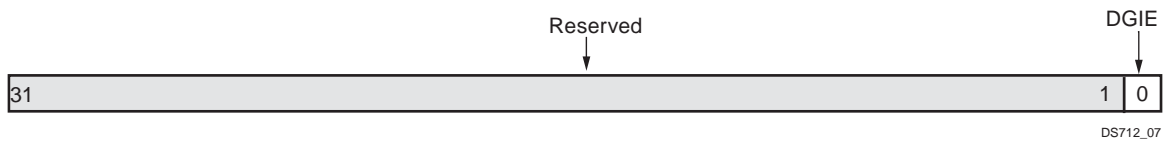


Figure 7: Device Global Interrupt Enable Register

Table 9: Device Global Interrupt Enable Register (DGIE) Bit Definitions

Bit(s)	Name	Core Access	Reset Value	Description
31-1	Reserved	N/A	0	Reserved
0	DGIE	R/W	'0'	Device Global Interrupt Enable Master Enable for routing Device Interrupt to the System Interrupt Controller. '1' = Enabled '0' = Disabled

Device Interrupt Enable Register (DIER)

The Device Interrupt Enable Register (DIER) is shown in Figure 8. It is a read/write register addressed at an offset 0x0C from base address C_S_AXI_CTRL_BASEADDR. The bit definitions of this register are as shown in Table 10. The Device Global Interrupt Enable Register provides the final enable/disable for the interrupt output to the processor and resides in the Register and Interrupt Module. This is a single bit read/write register as shown in Figure 8. The DIER bit definitions is shown in the Table 10.

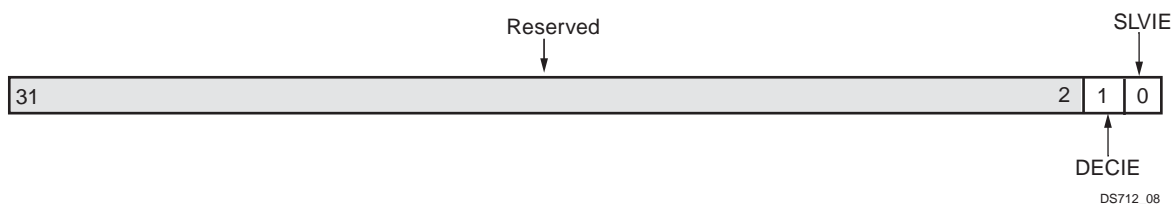


Figure 8: Device Interrupt Enable Register

Table 10: Device Interrupt Enable Register (DIER) Bit Definitions

Bit(s)	Name	Core Access	Reset Value	Description
31- 2	Reserved	N/A	0	Reserved
1	DECIE	R/W	'0'	DECERR Interrupt Enable Interrupt Enable bit for routing Decode error to the System Interrupt Controller. '1' = Interrupt will assert in response to DECERR '0' = Interrupt will not assert in response to DECERR
0	SLVIE	R/W	'0'	SLVERR Interrupt Enable Interrupt Enable bit for routing Slave error to the System Interrupt Controller. '1' = Interrupt will assert in response to SLVERR '0' = Interrupt will not assert in response to SLVERR

Address Decoding and Memory Mapping

As AXI to PLBv46 Bridge will be a P2P interface on interconnect - address decoding is not implemented for the port that gets translated to PLBv46. Hence, it responds to all addresses presented.

The address ranges specified by the pair of the parameters C_S_AXI_BASEADDR, C_S_AXI_HIGHADDR and C_S_AXI_CTRL_BASEADDR, C_S_AXI_CTRL_HIGHADDR are to inform the interconnect about the address map of the PLB sub-system and internal register of the bridge.

PLBv46 Remote Slave Rearbitration

The AXI to PLBv46 Bridge does not decode PLB_Mrearbitrate - the request on PLB will be valid till PLB_MAddrAck or PLB_MTimeout is asserted.

Clocking

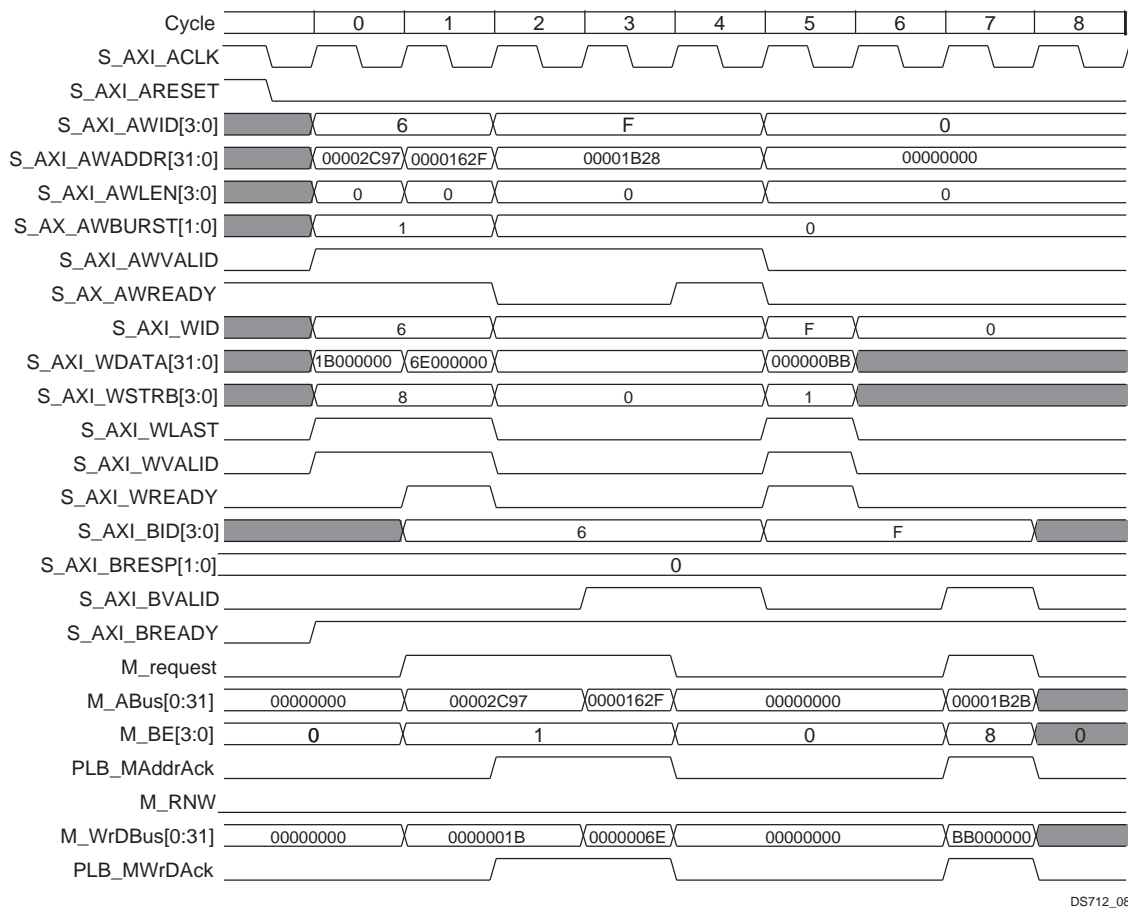
The AXI to PLBv46 Bridge has single clock source. Supports 1:1 (AXI:PLB) clock ratio.

Reset

The AXI to PLBv46 Bridge has a single reset source. As long as the whole system (or at least PLB/ AXI sides) is reset in the same clock cycle and released in the same clock cycle will not have any issue.

Timing Diagrams

FIXED/INCR Single Write



DS712_08

Figure 9: Single Write

FIXED/INCR Single Read

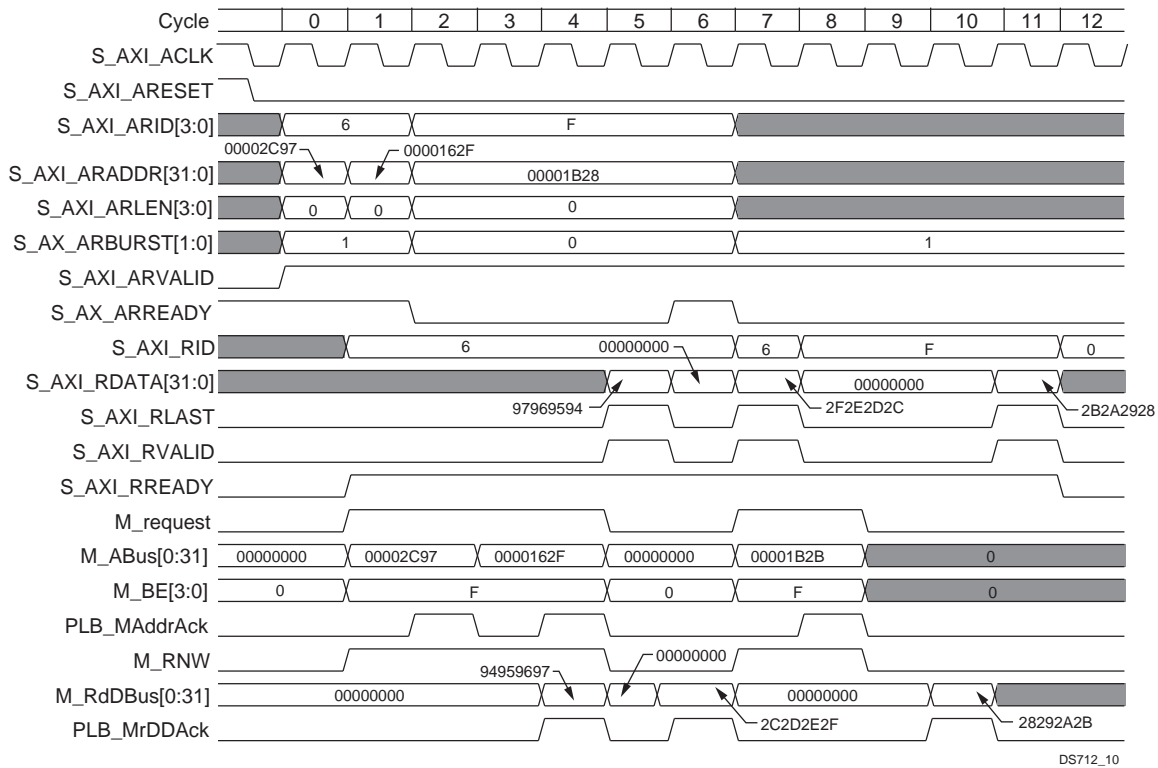


Figure 10: Single Read

FIXED/INCR Single Read Write

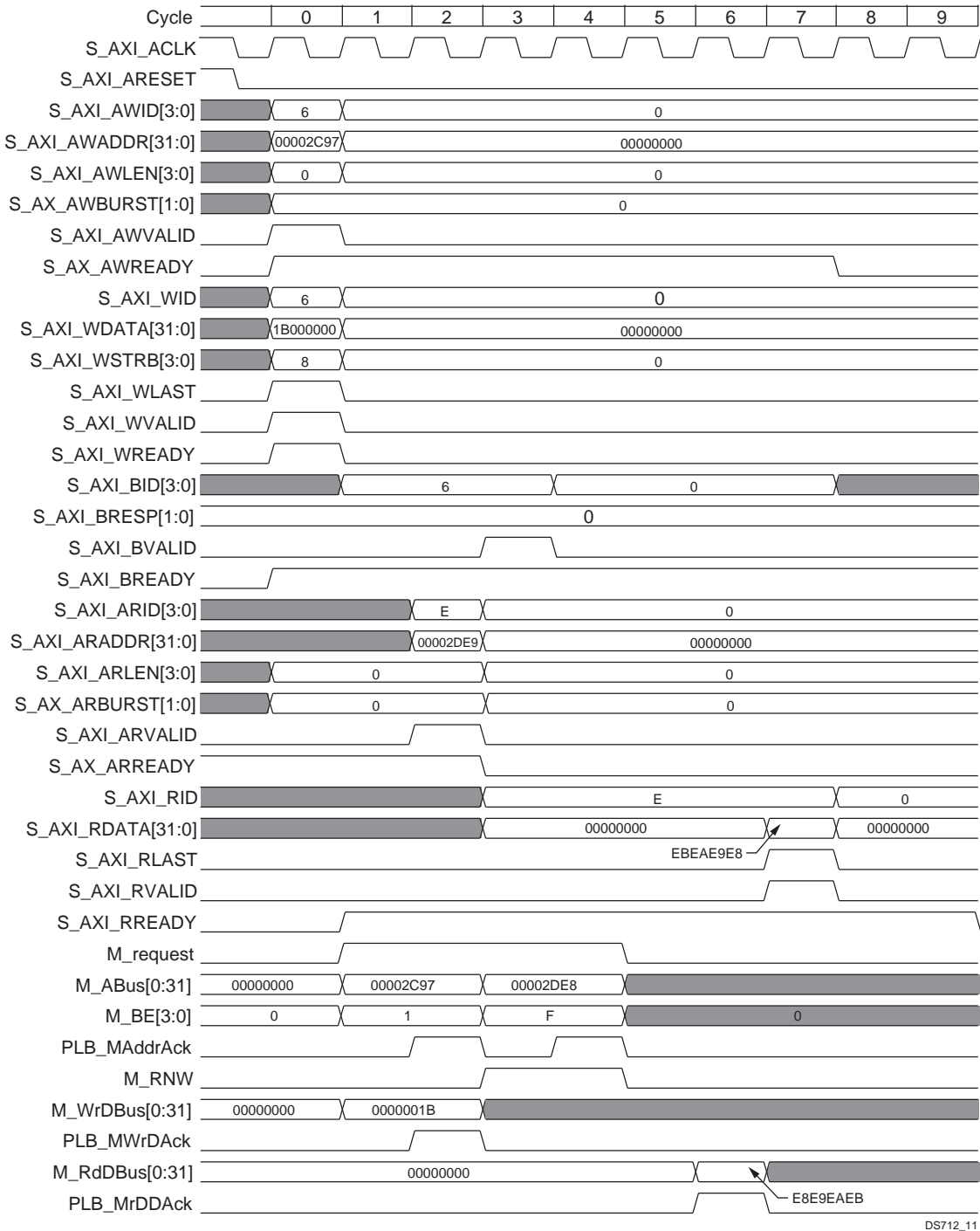
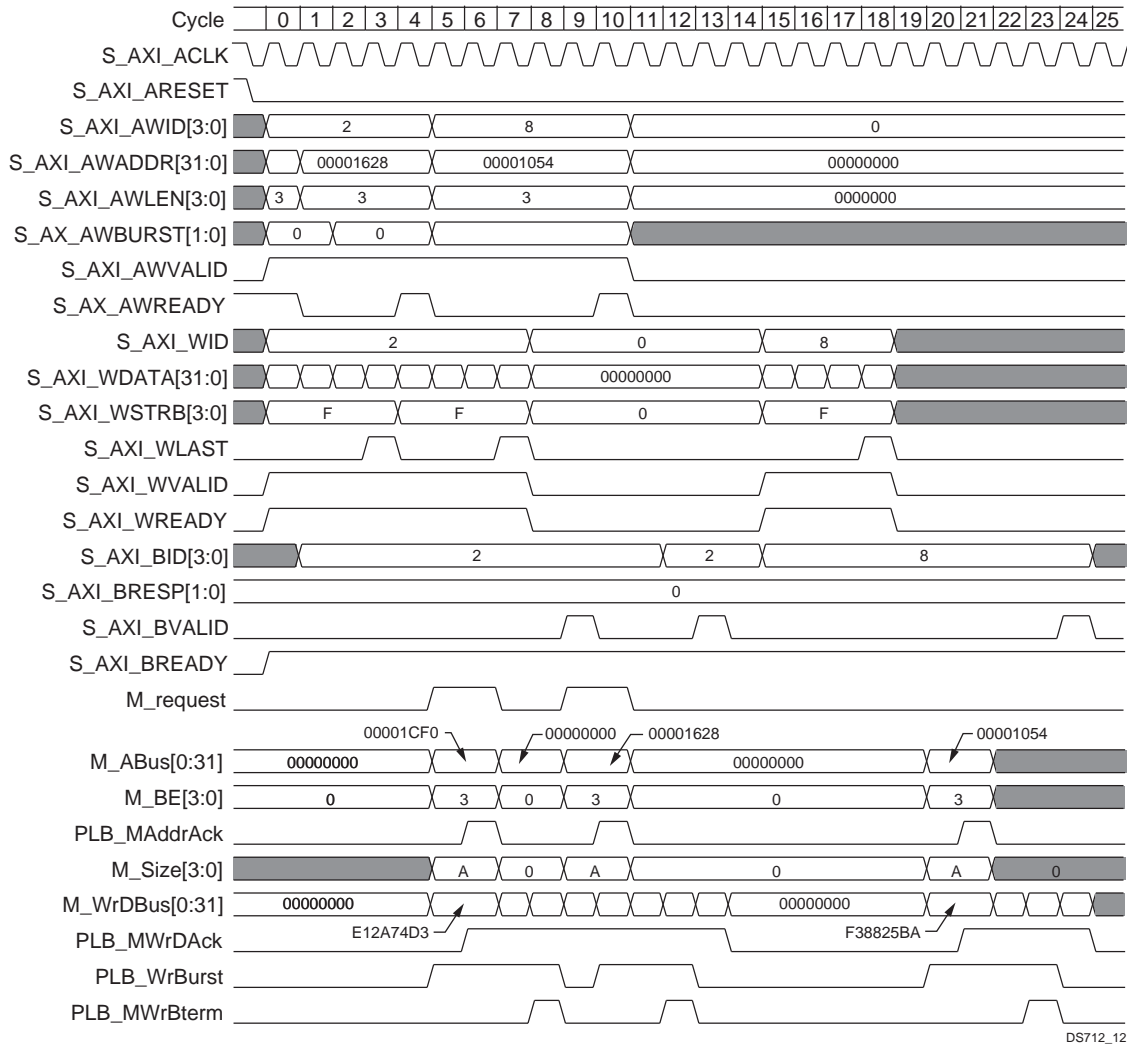
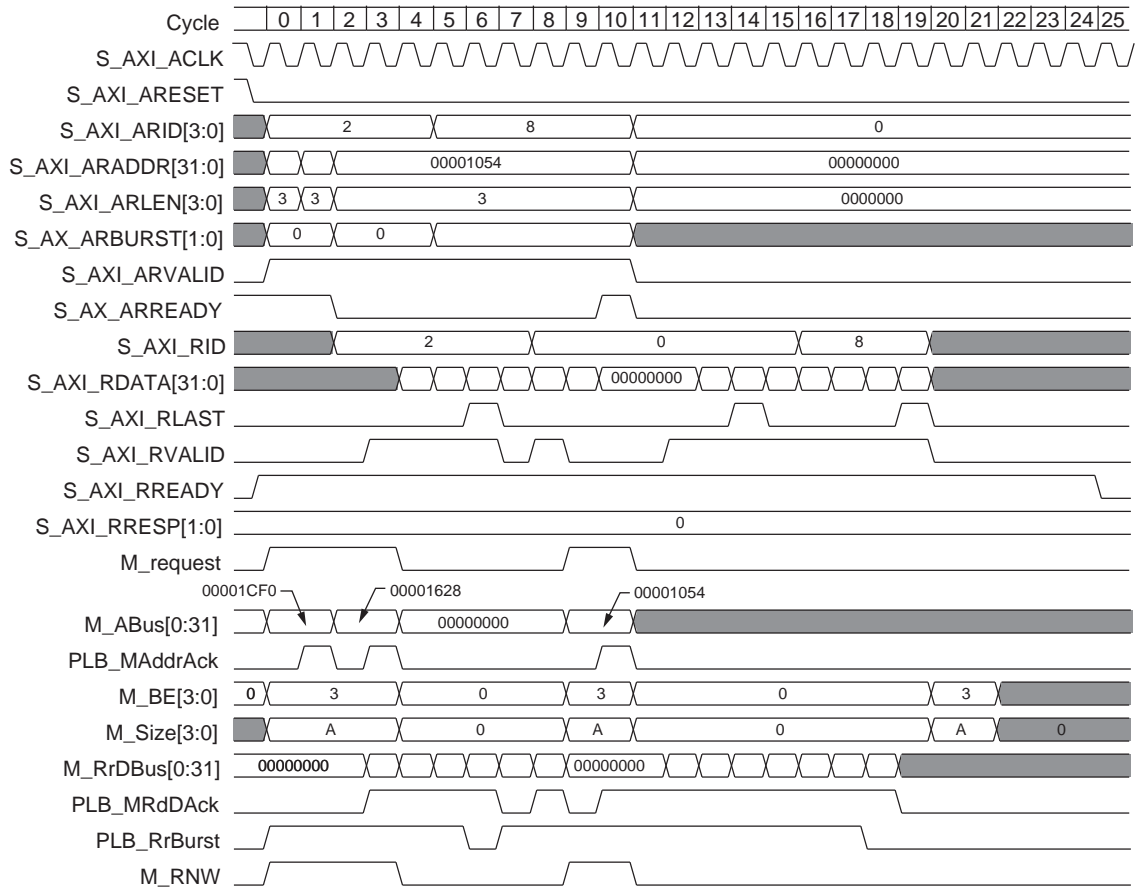


Figure 11: Single Read Write

FIXED/INCR 4 word Burst Write



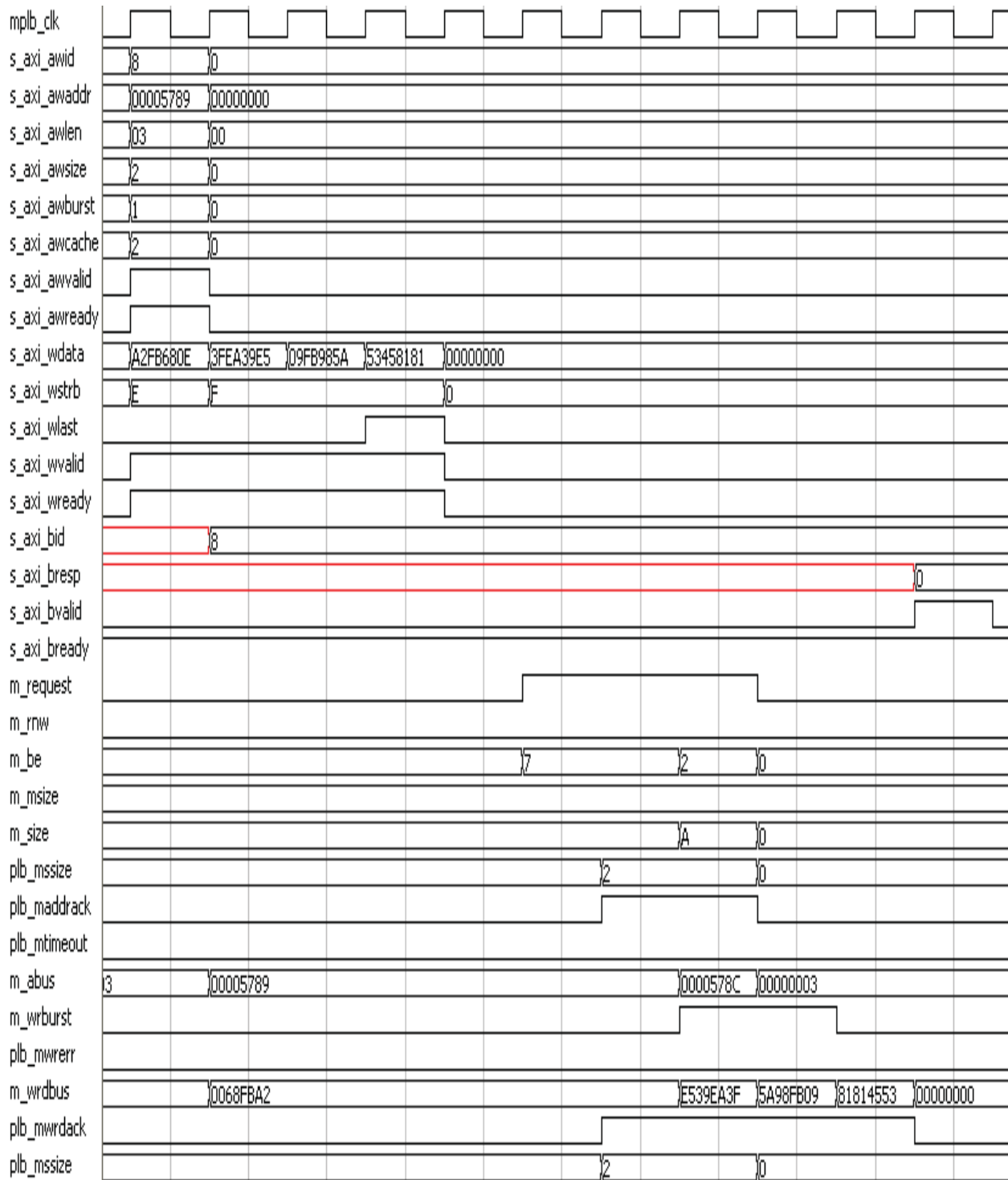
FIXED/INCR 4 word Burst Read



DS712_13

Figure 13: Burst Read

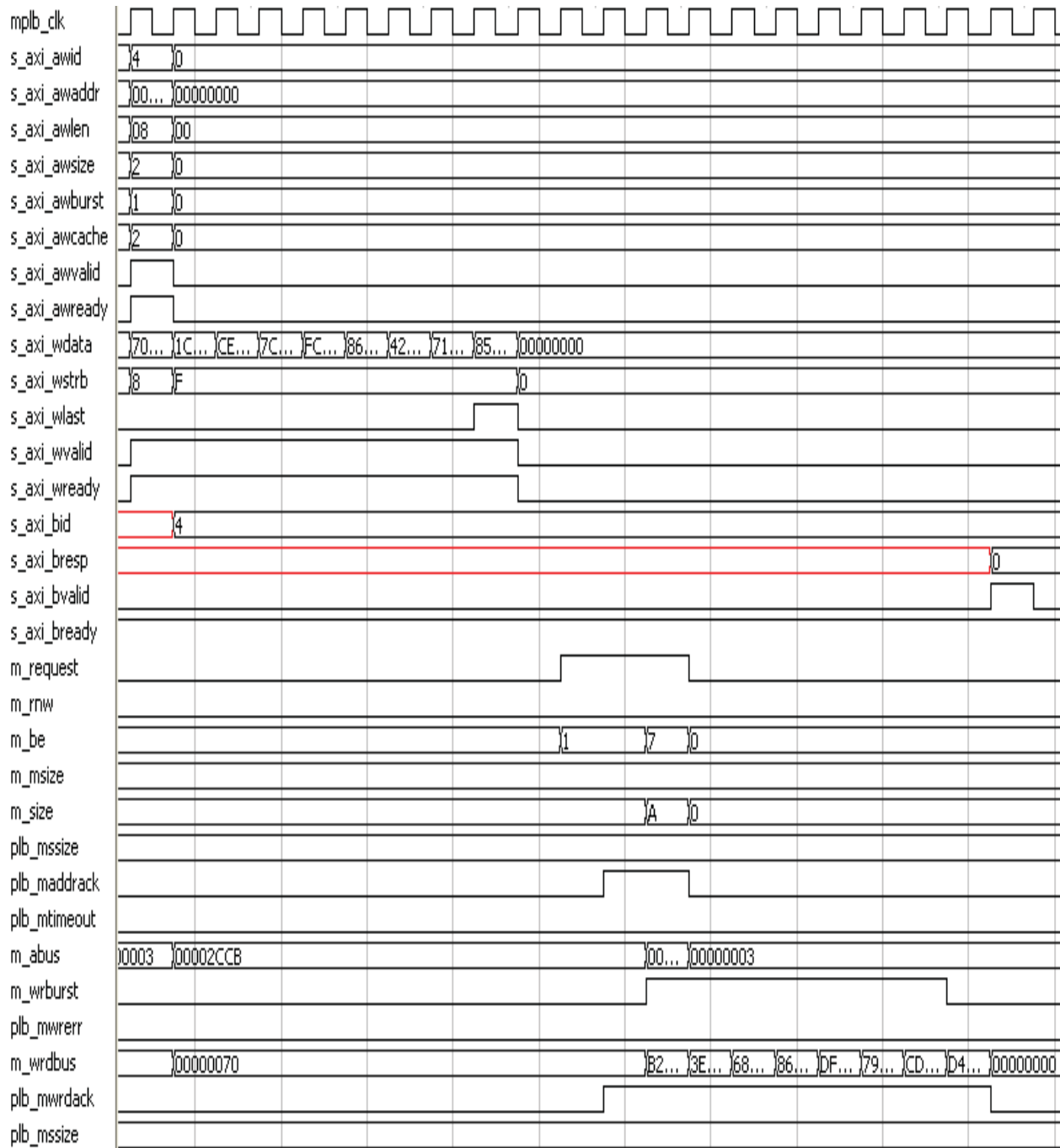
INCR Write 4 Beat



DS712_14

Figure 14: INCR Write 4 Beat

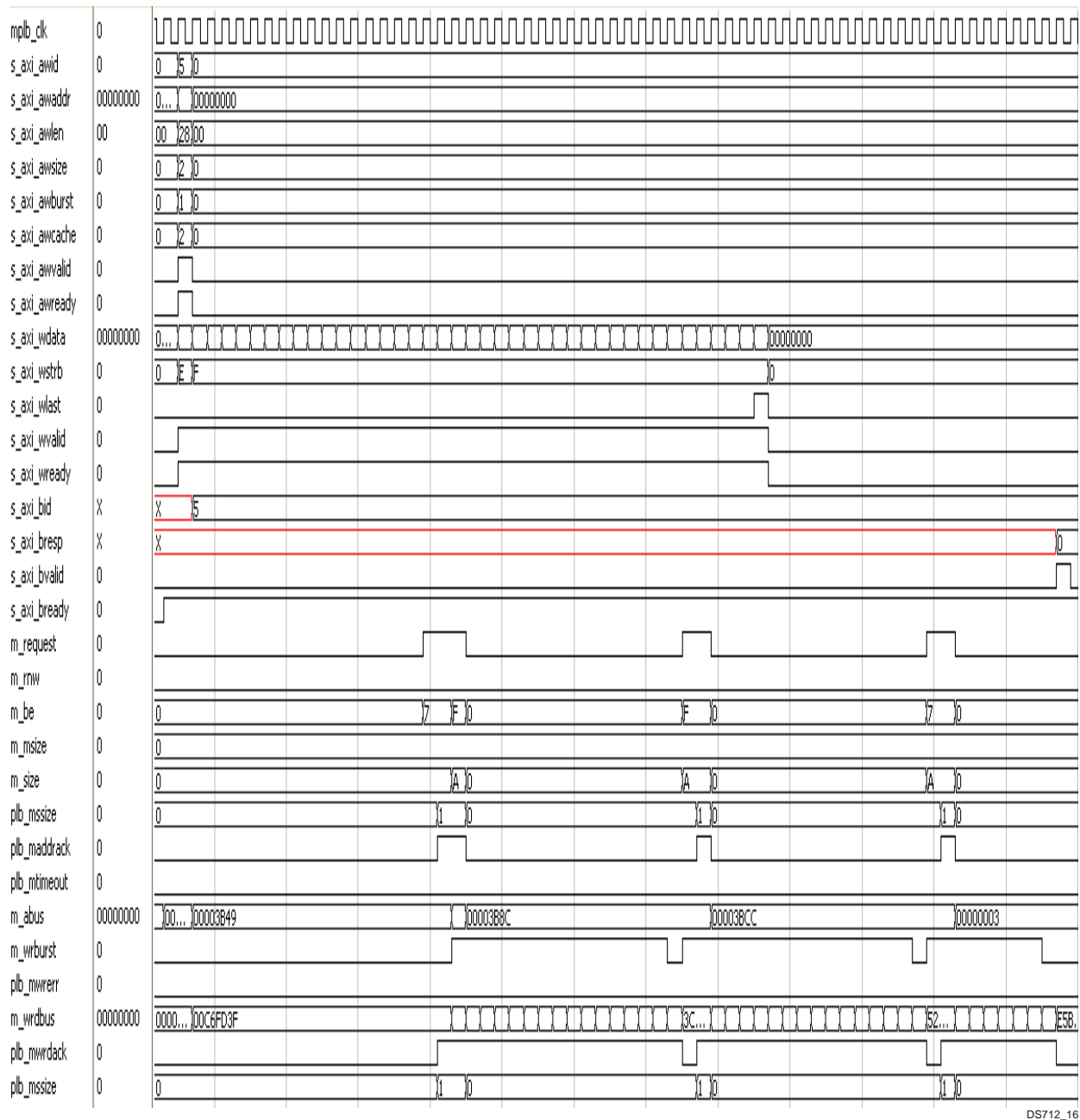
INCR Write 9 Beat



DS712_15

Figure 15: INCR Wr 9 Beat

INCR Write 29 Beat



DS712_16

Figure 16: INCR Wr 29 Beat

[illegible]

Figure 17: INCR Wr 28 Half Word

INCR Write 17 bytes

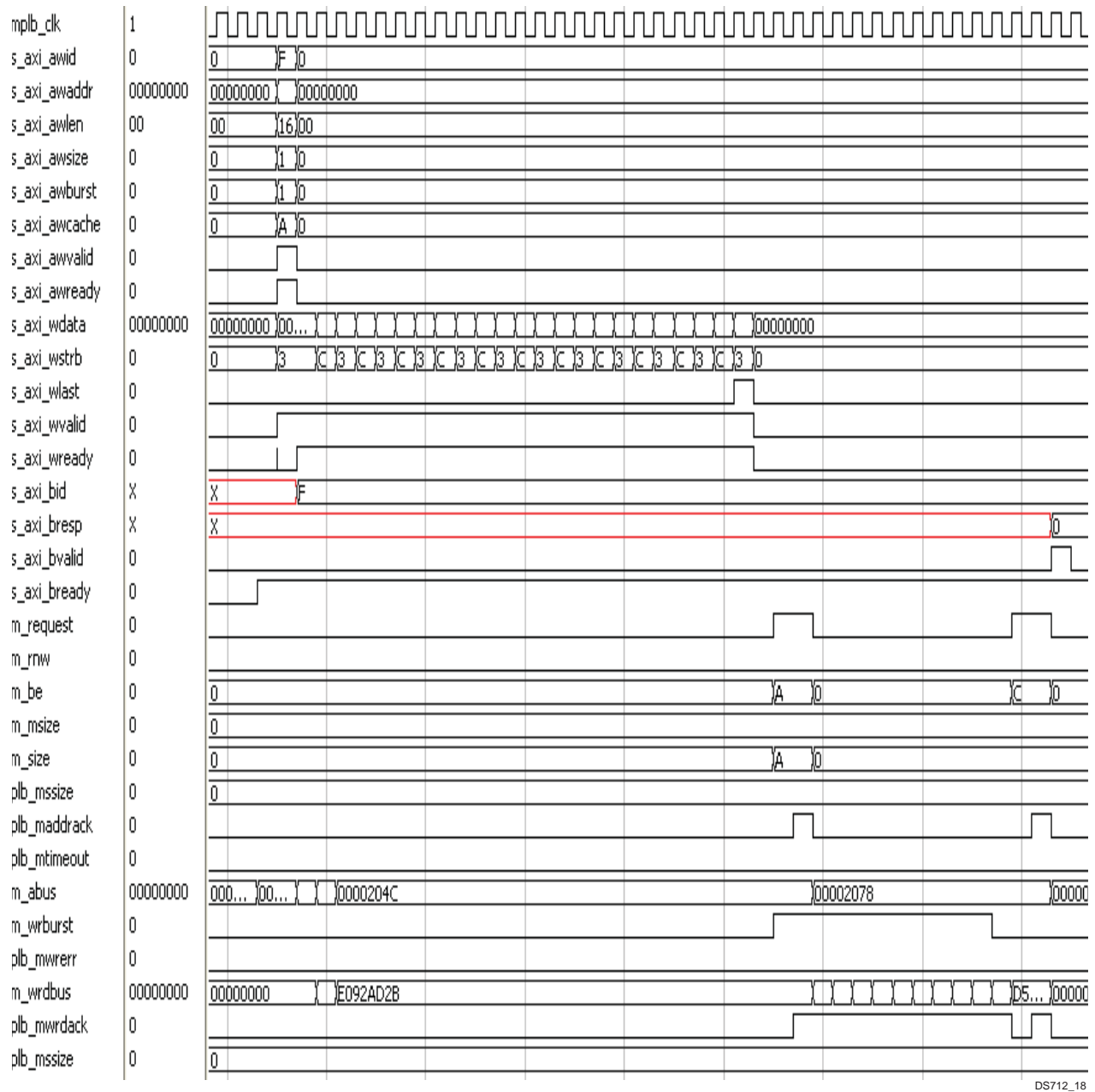
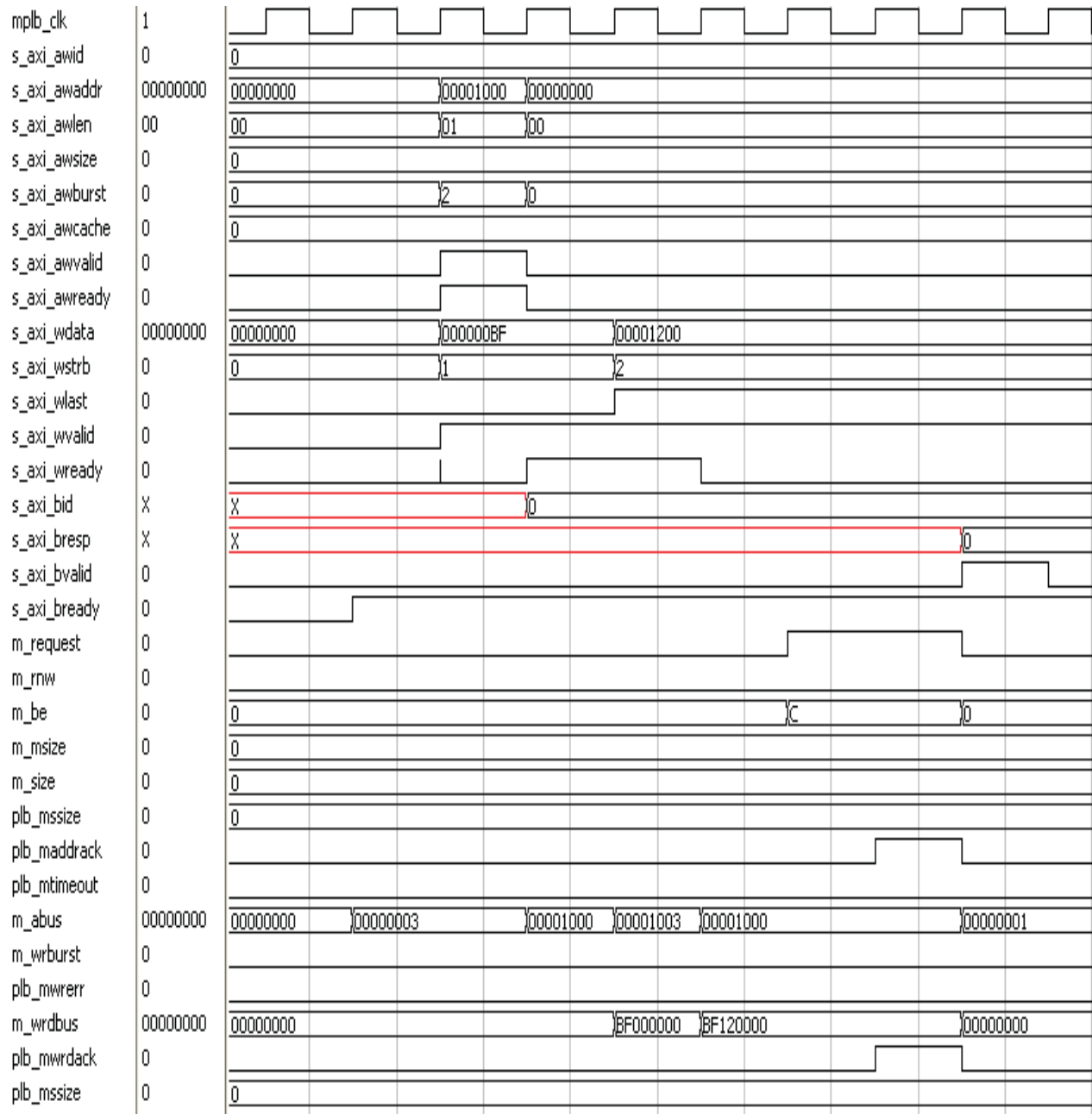


Figure 18: INCR Write 17 byte

DS712_18

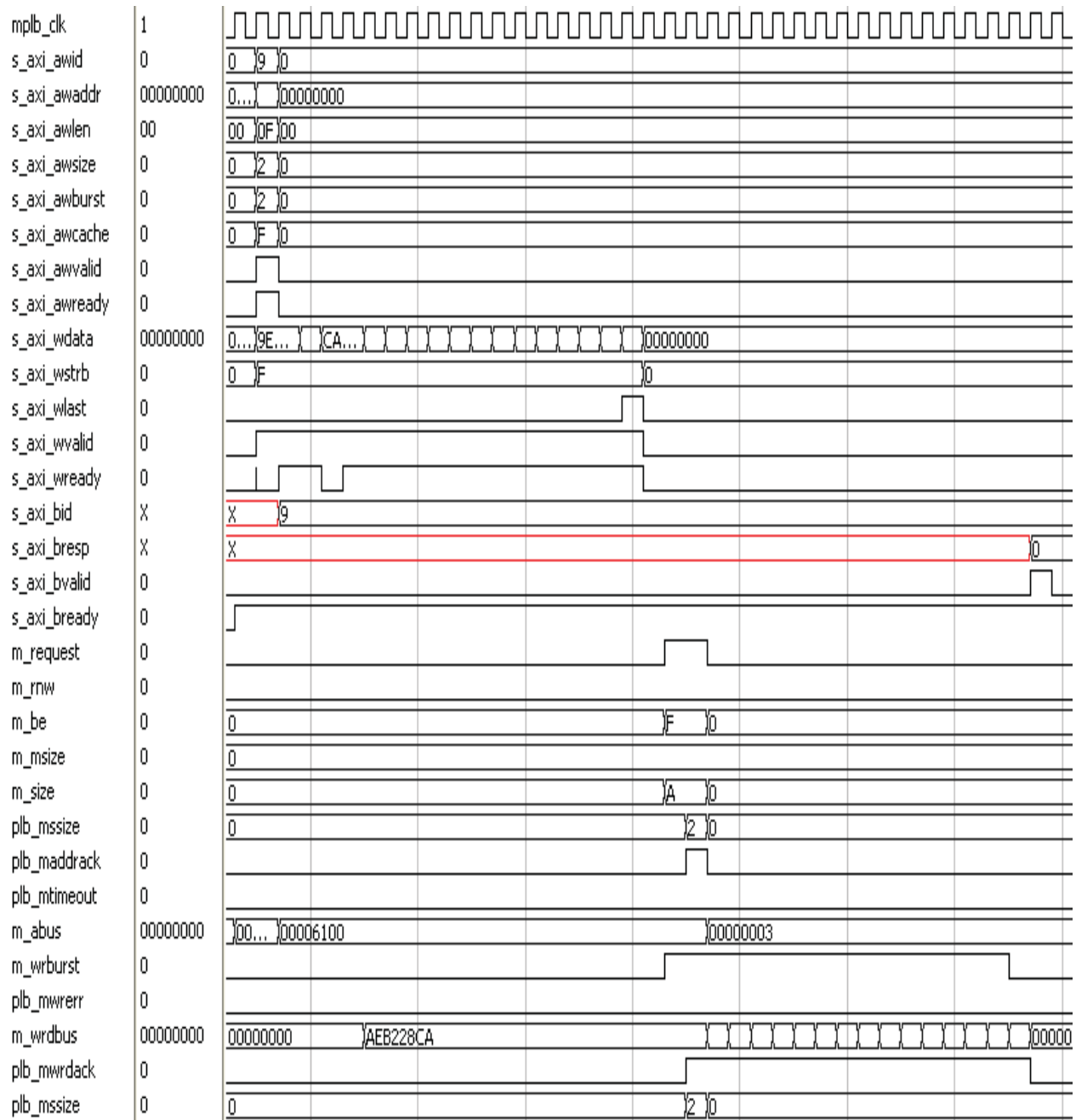
WRAP Write 2 byte



DS712_19

Figure 19: WRAP Write 2 byte

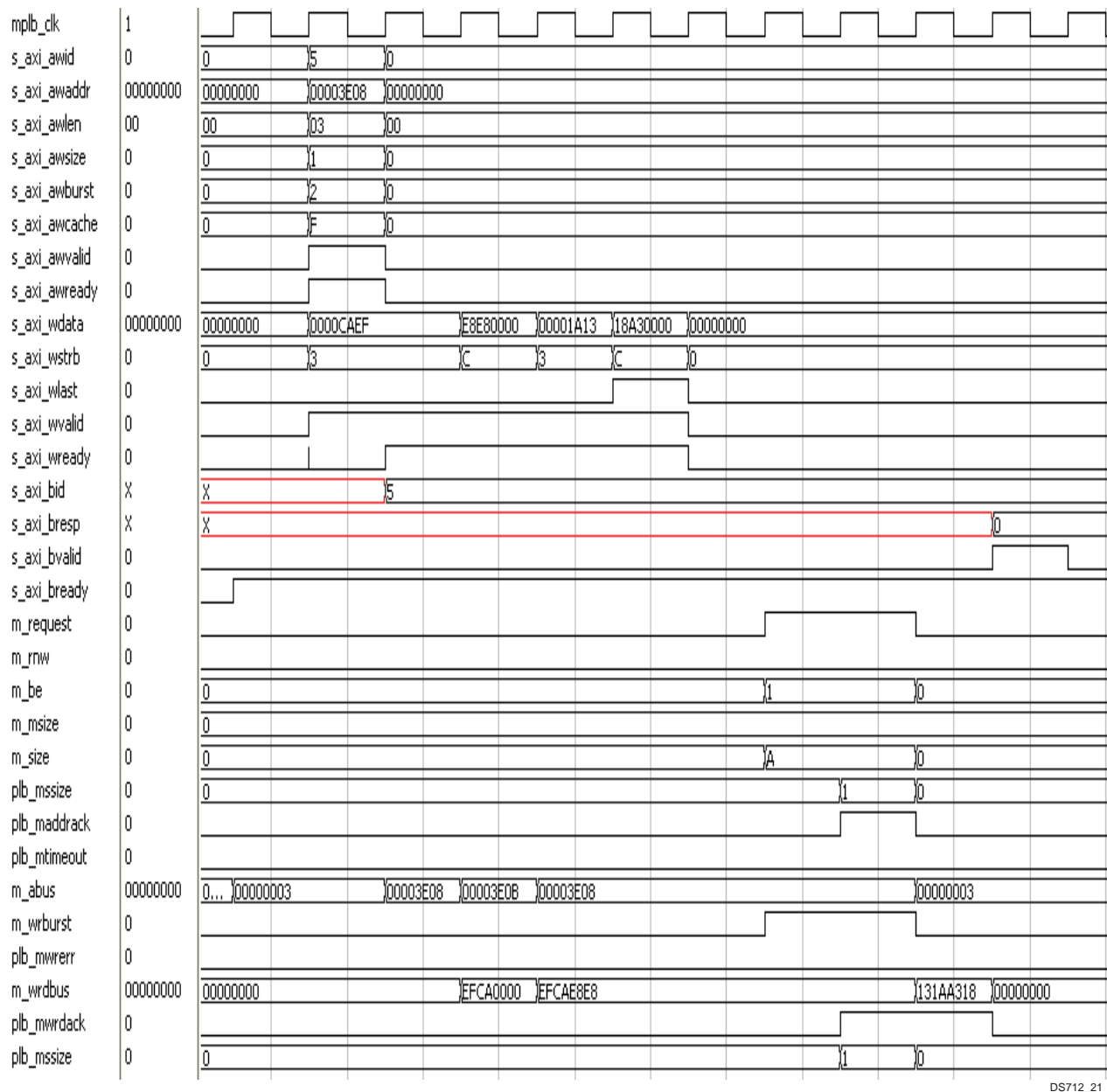
WRAP 16 Beats



DS712_20

Figure 20: WRAP 16 Beats

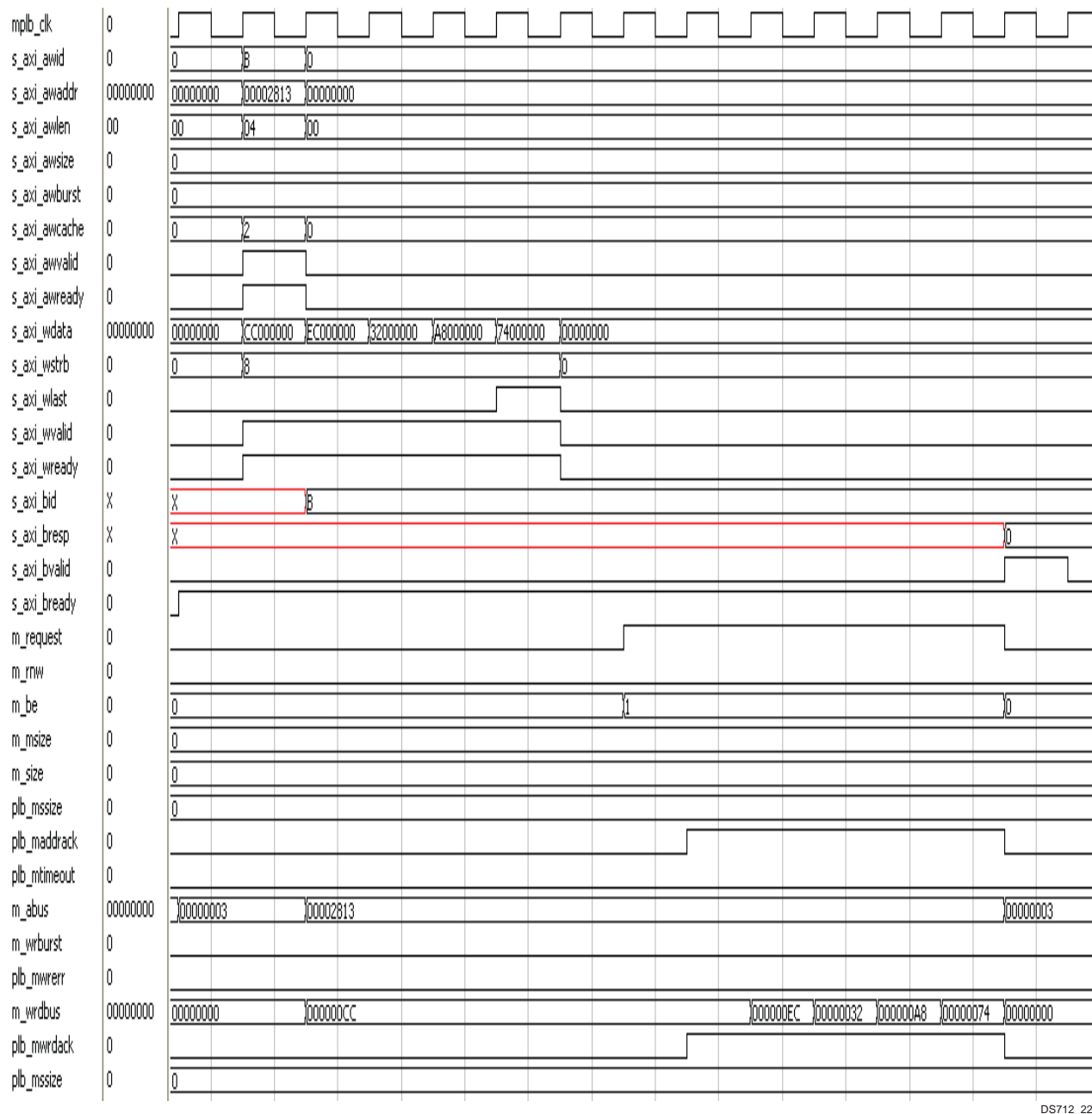
WRAP 4 Half Word



DS712_21

Figure 21: WRAP 4 Half Word

FIXED Write 5 byte



DS712_22

Figure 22: FIXED Write 5 byte

FIXED Write 13 Beat

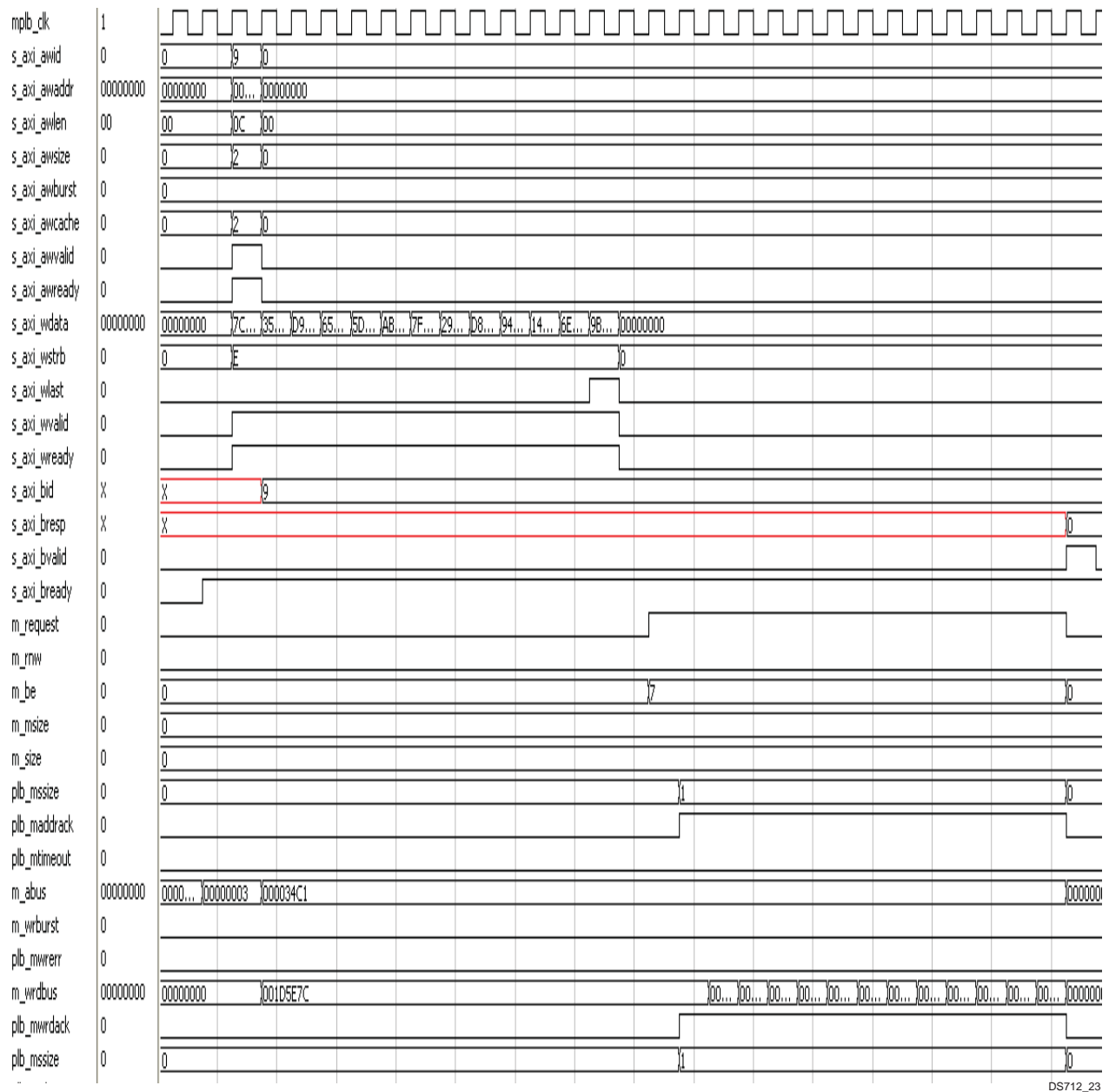
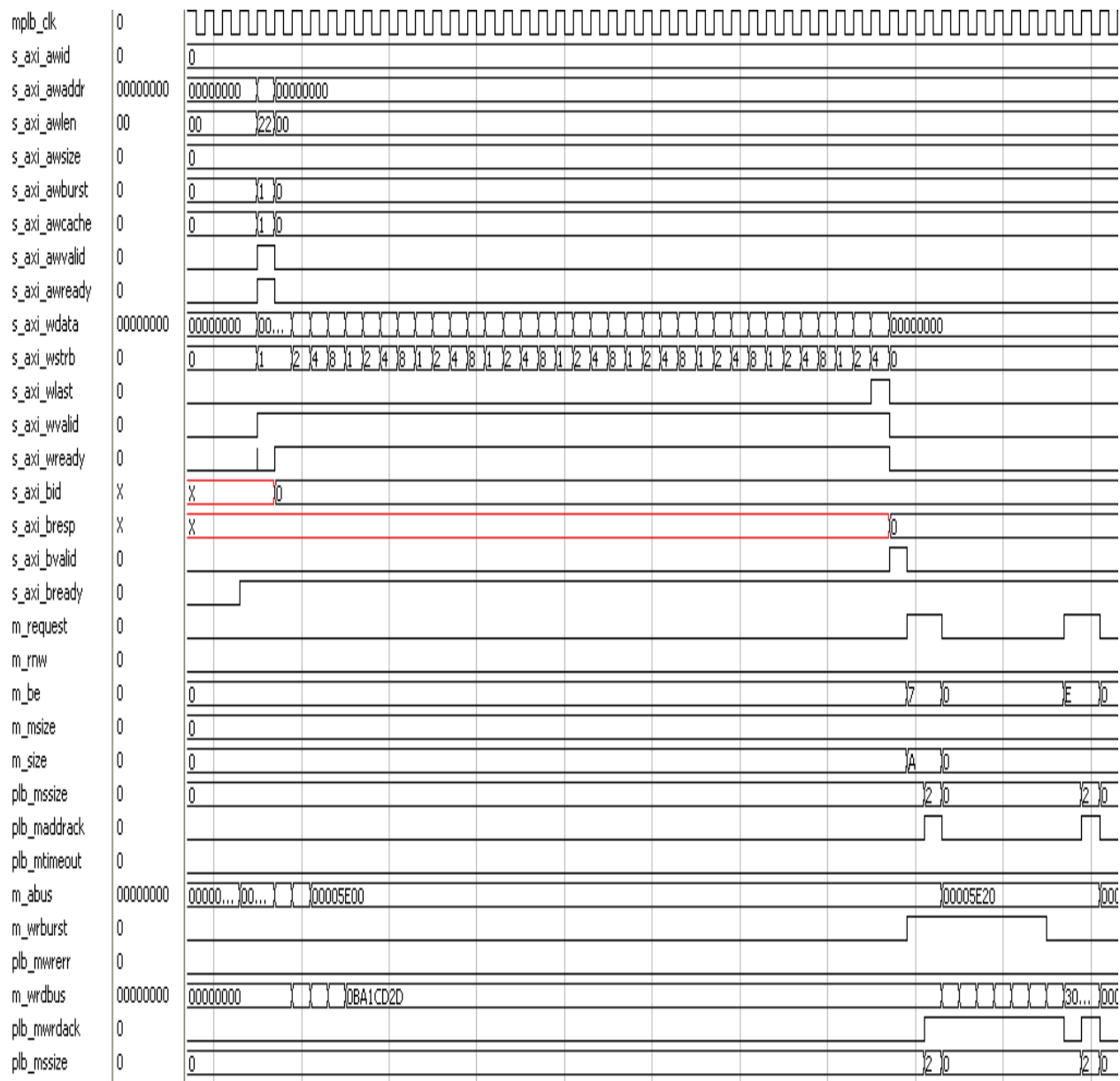


Figure 23: **FIXED Write 13 Beat**

INCR Write 23 bytes - Bufferable



DS712_24

Figure 24: INCR Write 23 bytes - Bufferable

INCR Write 23 bytes - bufferable Error Interrupt

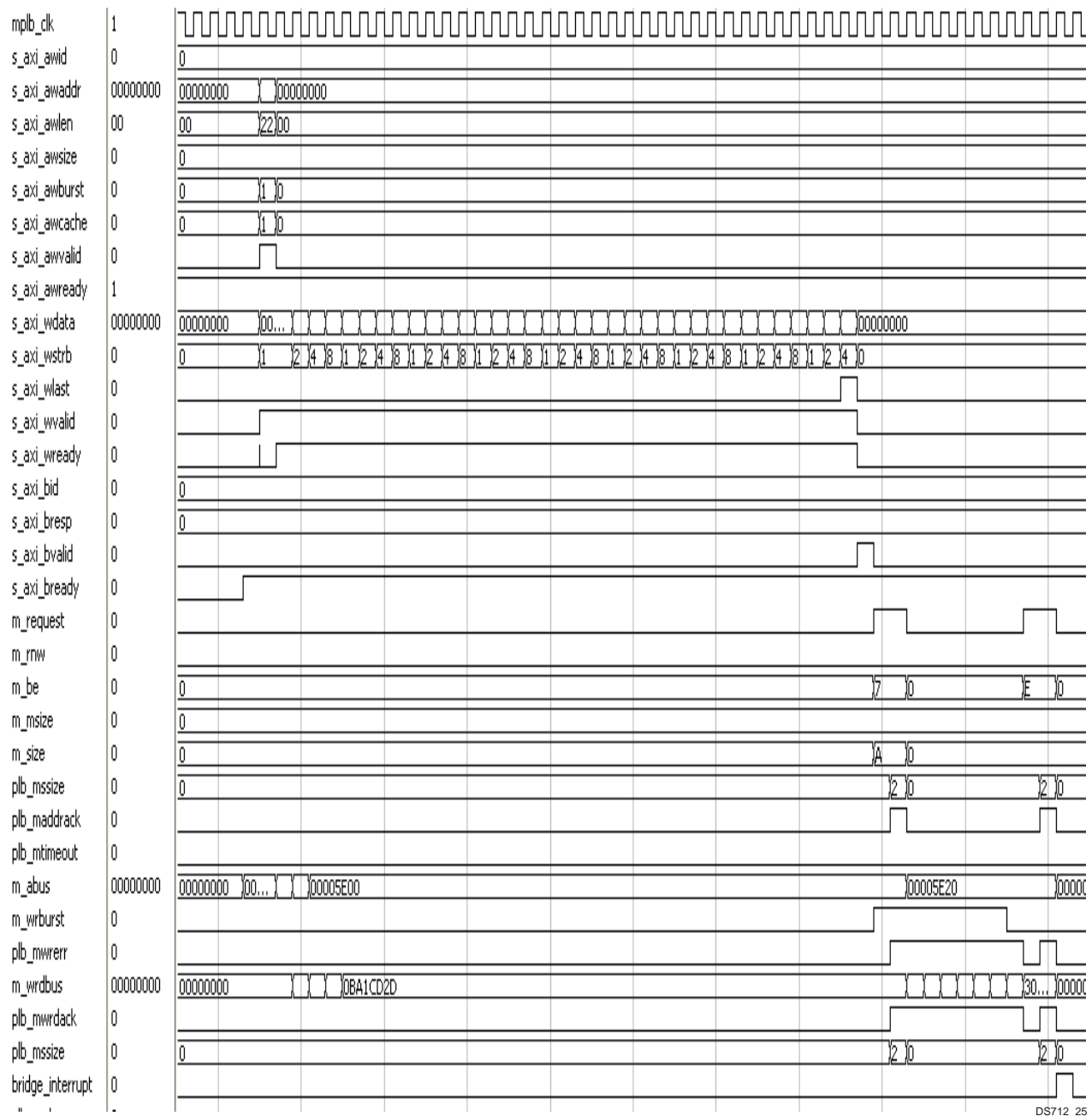
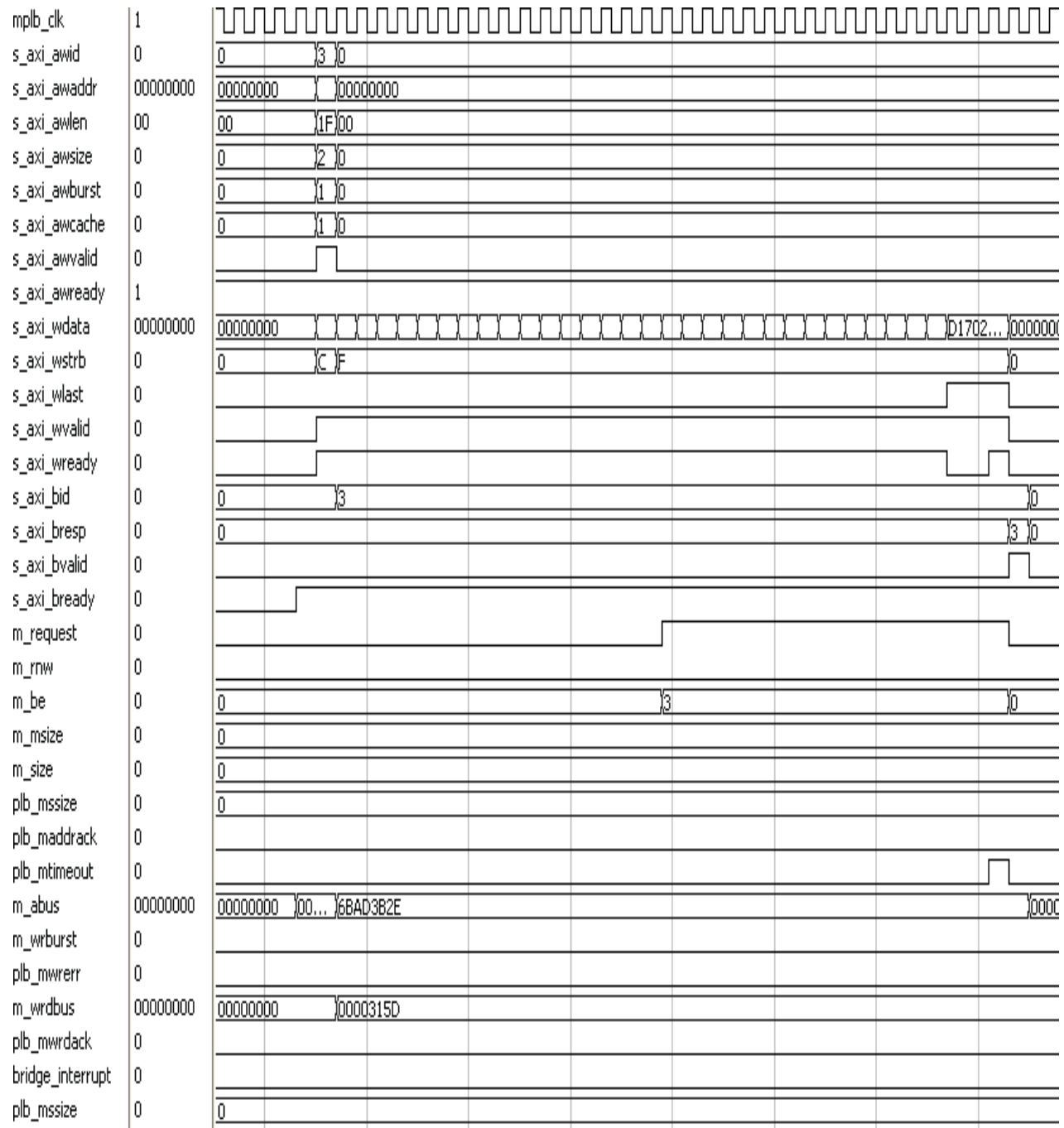


Figure 25: INCR Write 23 bytes - bufferable Error Interrupt

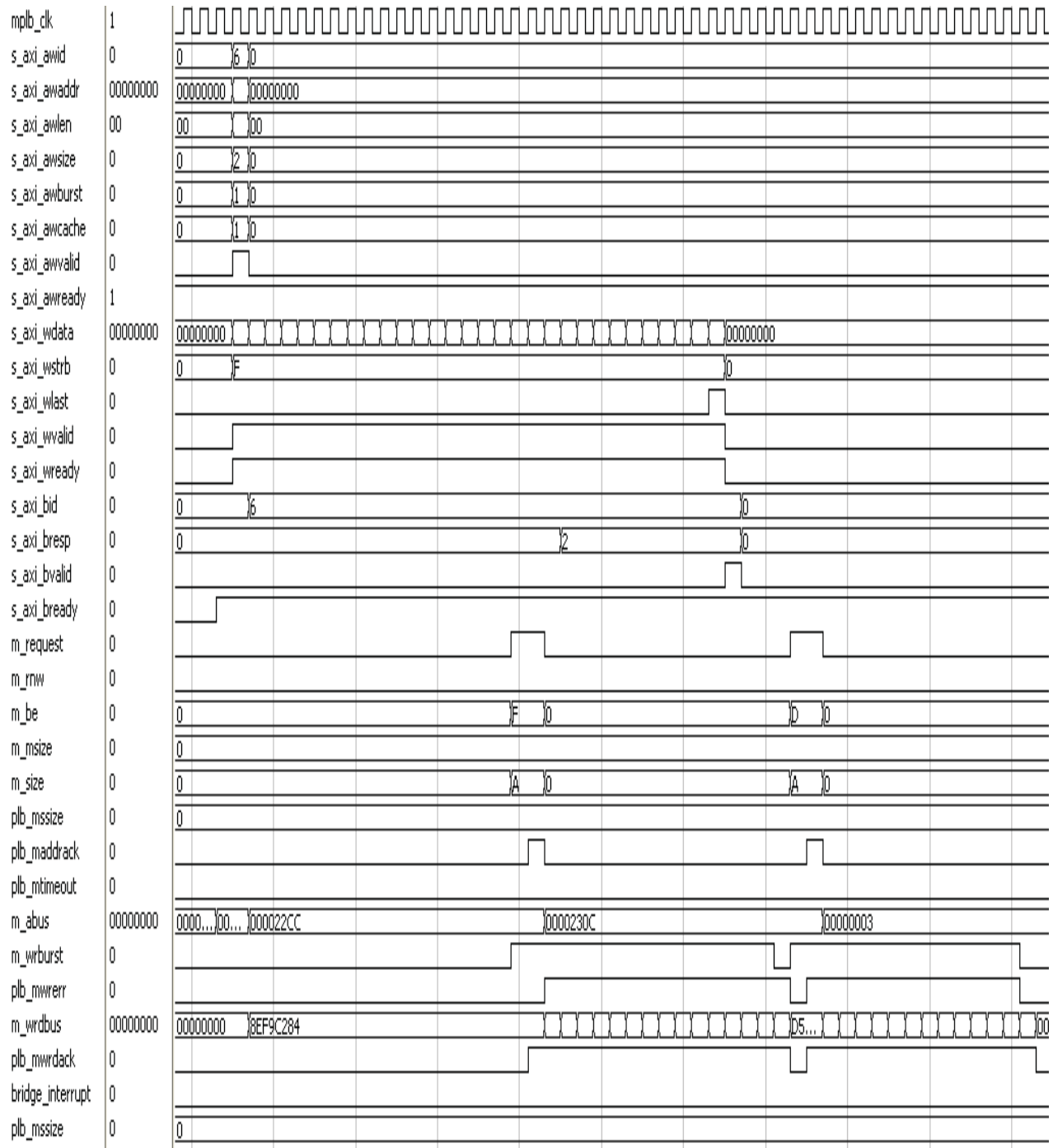
INCR Write 20 beat - Bufferable Timeout



DS712_26

Figure 26: INCR Write 20 beat - Bufferable Timeout

INCR Write 1E beat - bufferable Error



DS712_27

Figure 27: INCR Write 1E beat - Bufferable Error

INCR Write 8 beat - bufferable Timeout Interrupt

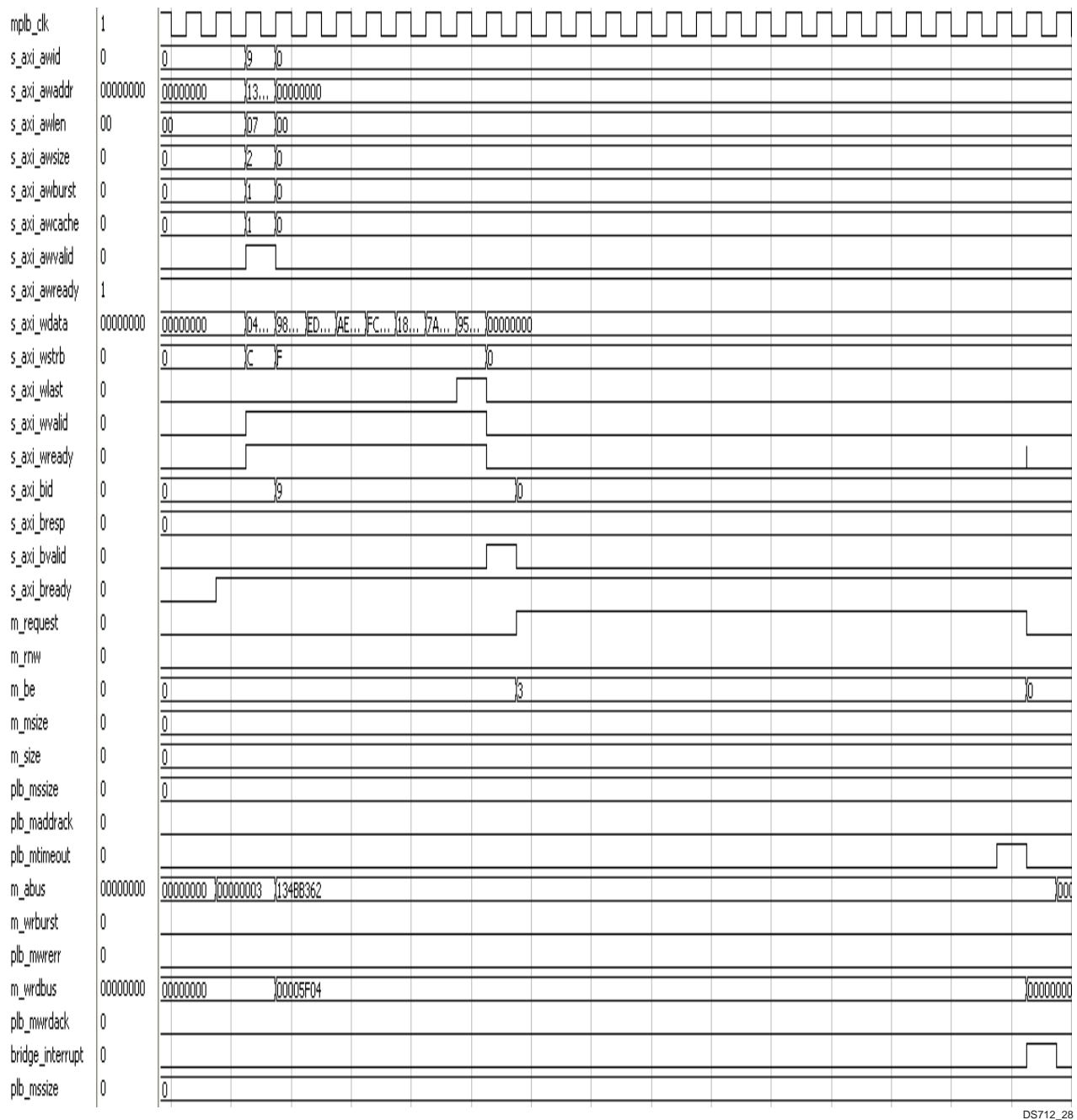
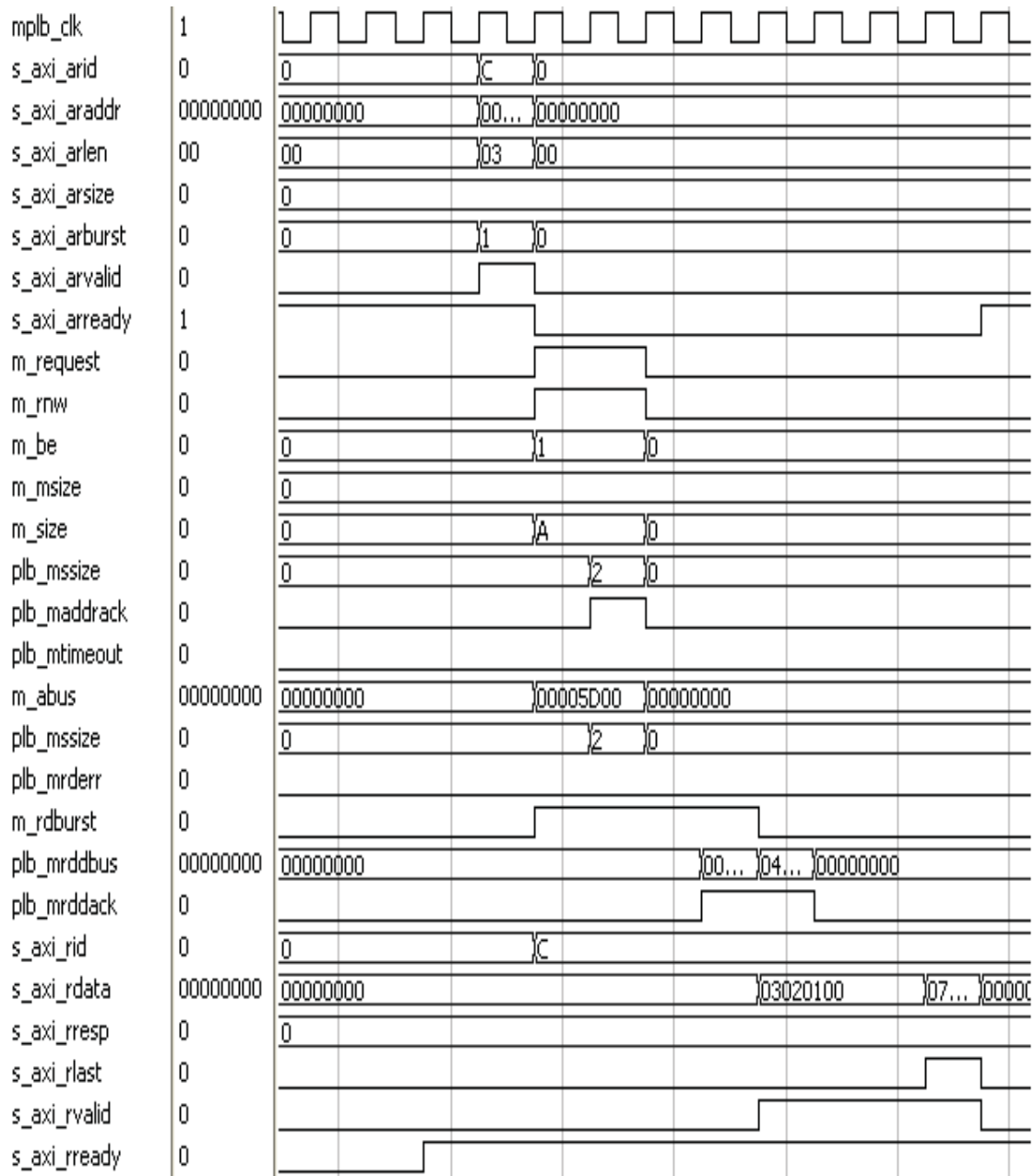


Figure 28: INCR Write 8 beat - Bufferable Timeout Interrupt

INCR Read 4 byte



DS712_29

Figure 29: INCR Read 4 byte

INCR Read 1D beat

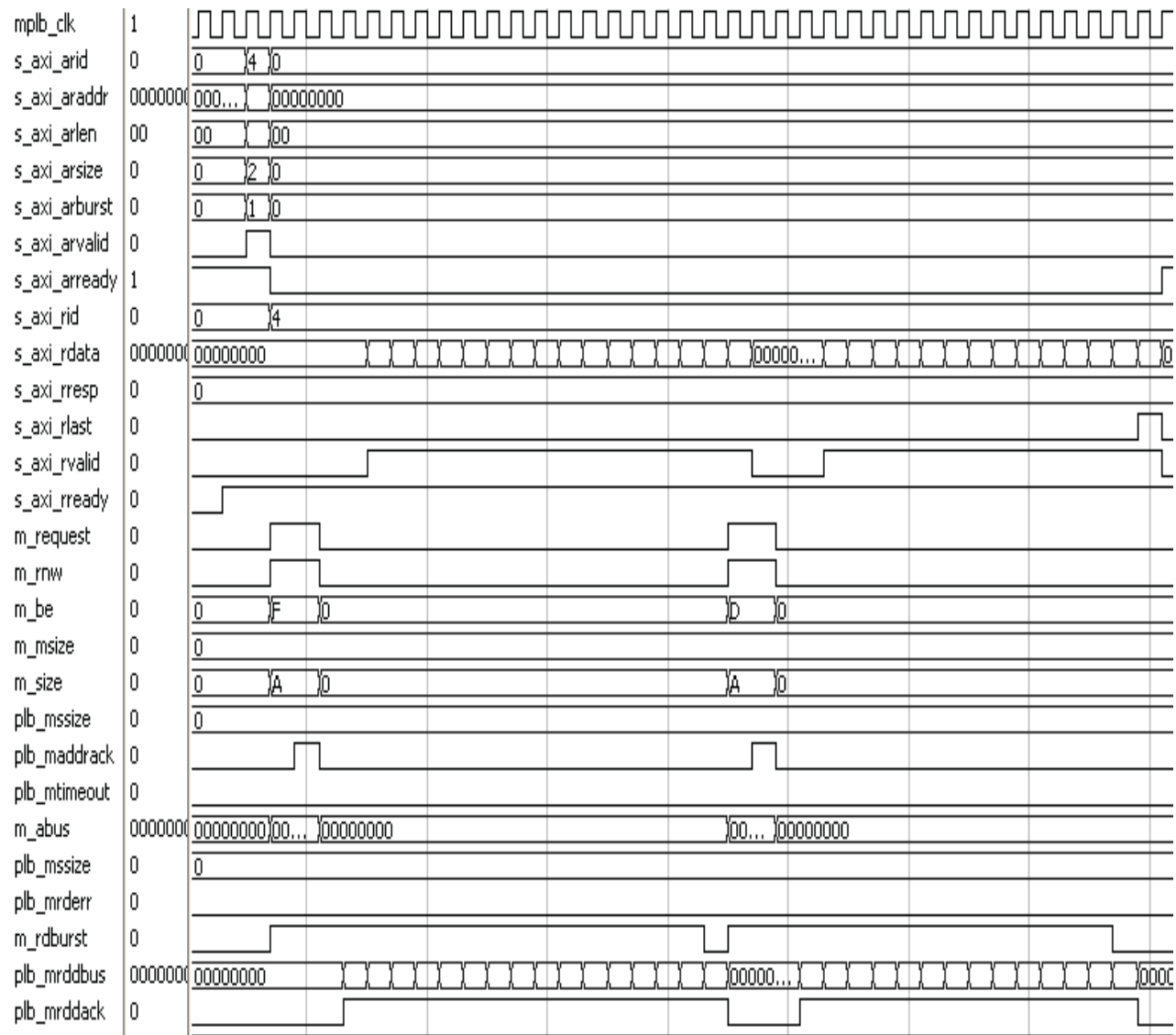
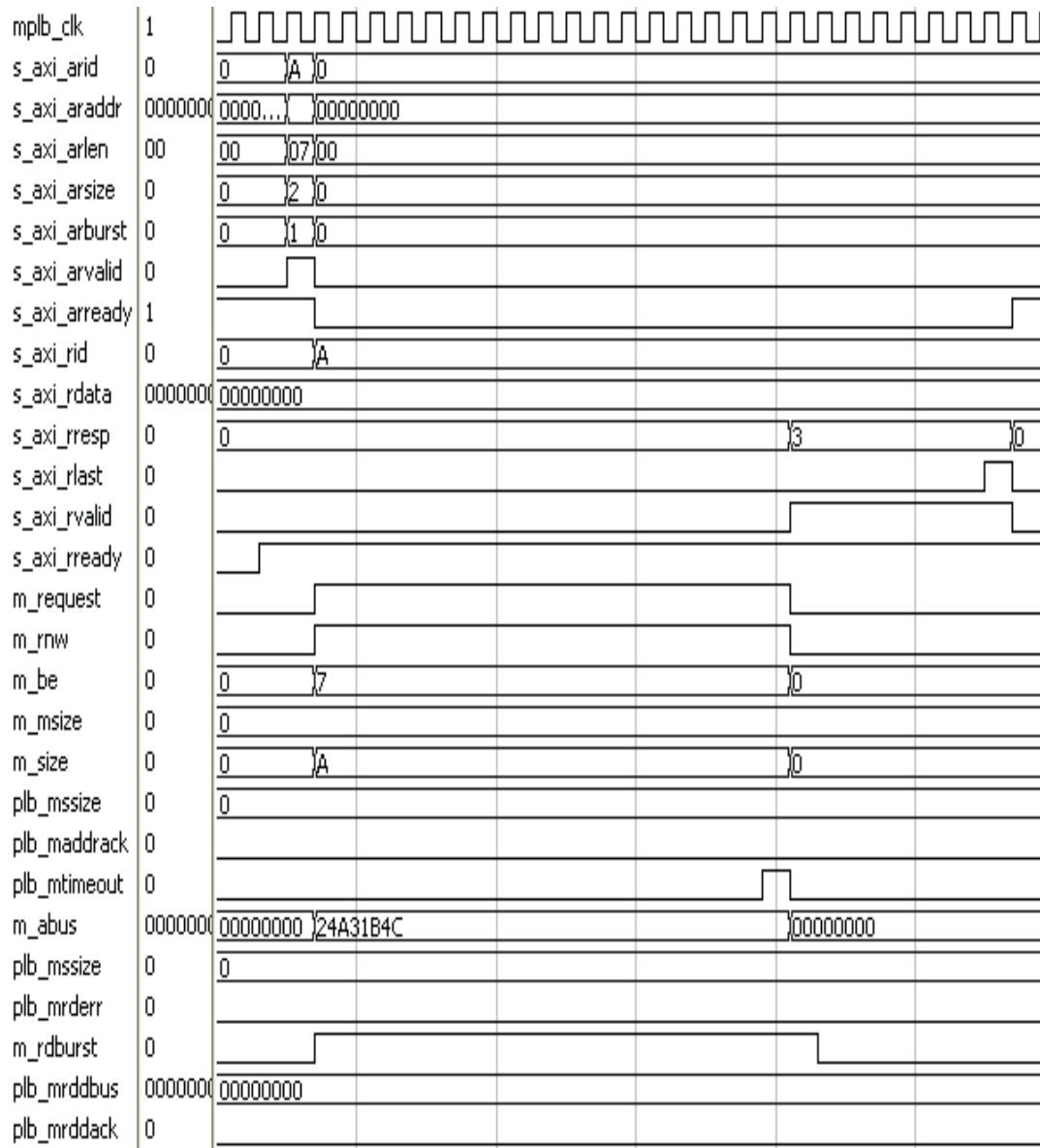


Figure 30: INCR Read 1D Beat

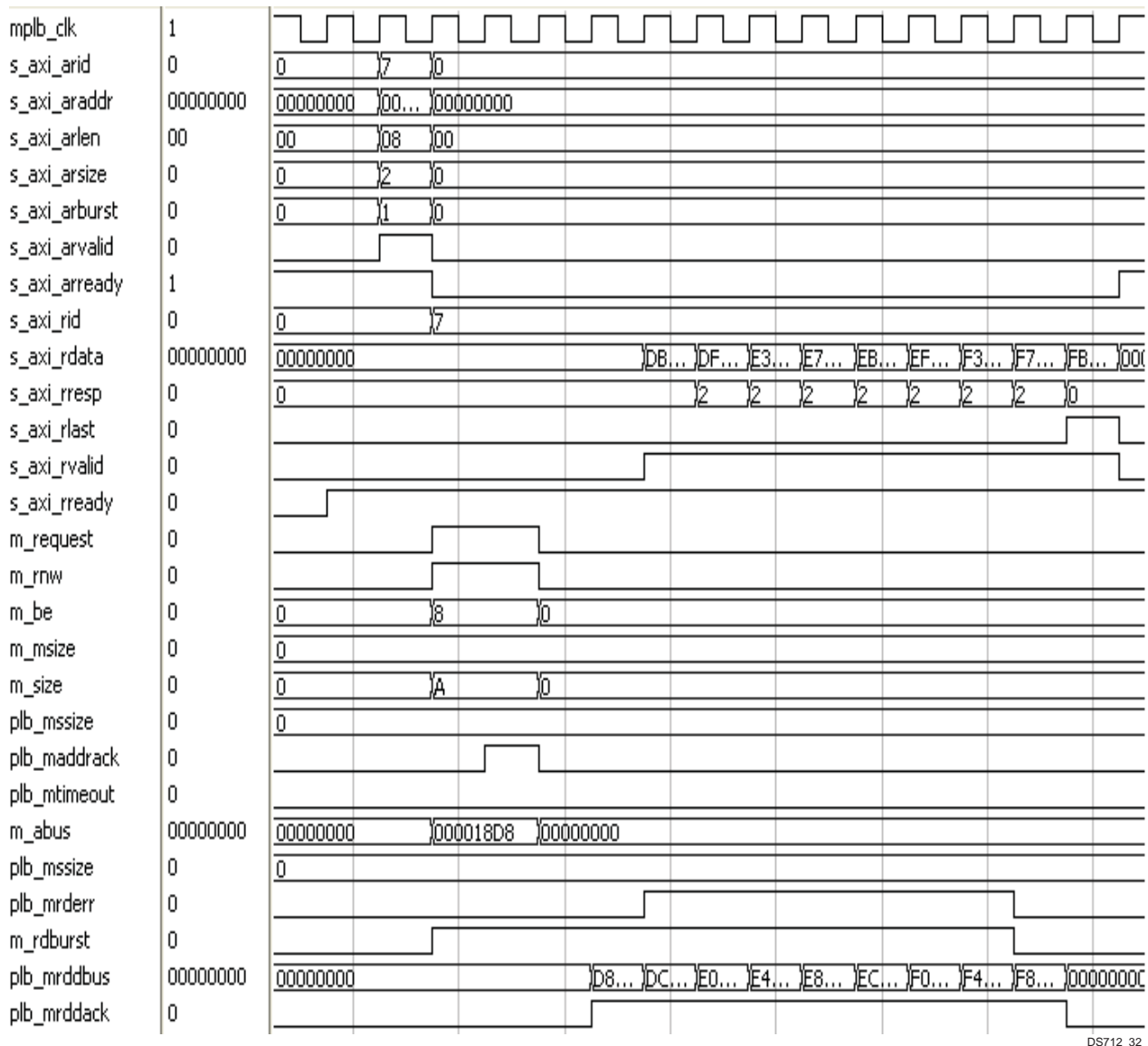
INCR Read 7 Beat Timeout



DS712_31

Figure 31: INCR Read 7 Beat Timeout

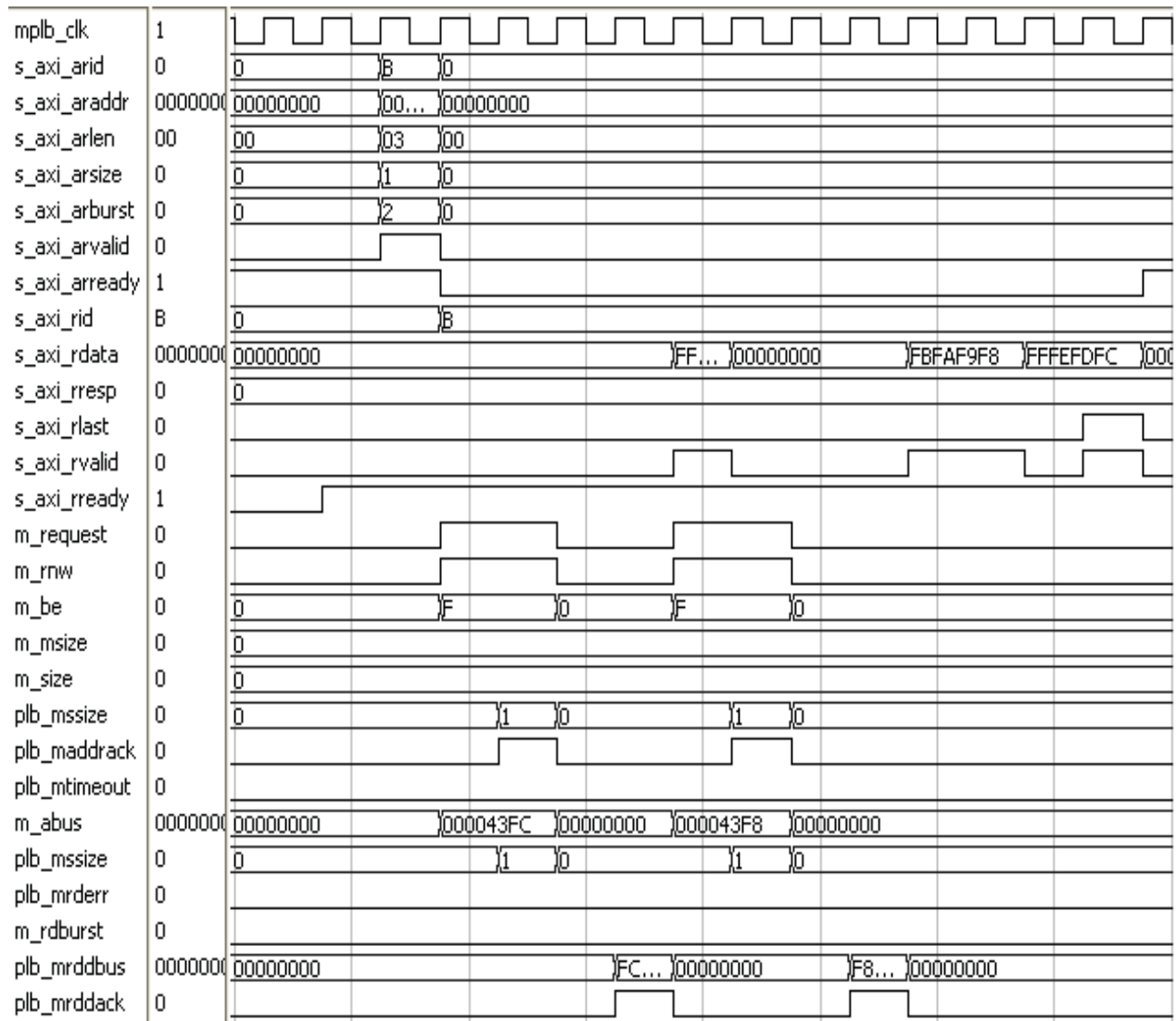
INCR Read 8-Beat Error



DS712_32

Figure 32: INCR Read 8-Beat Error

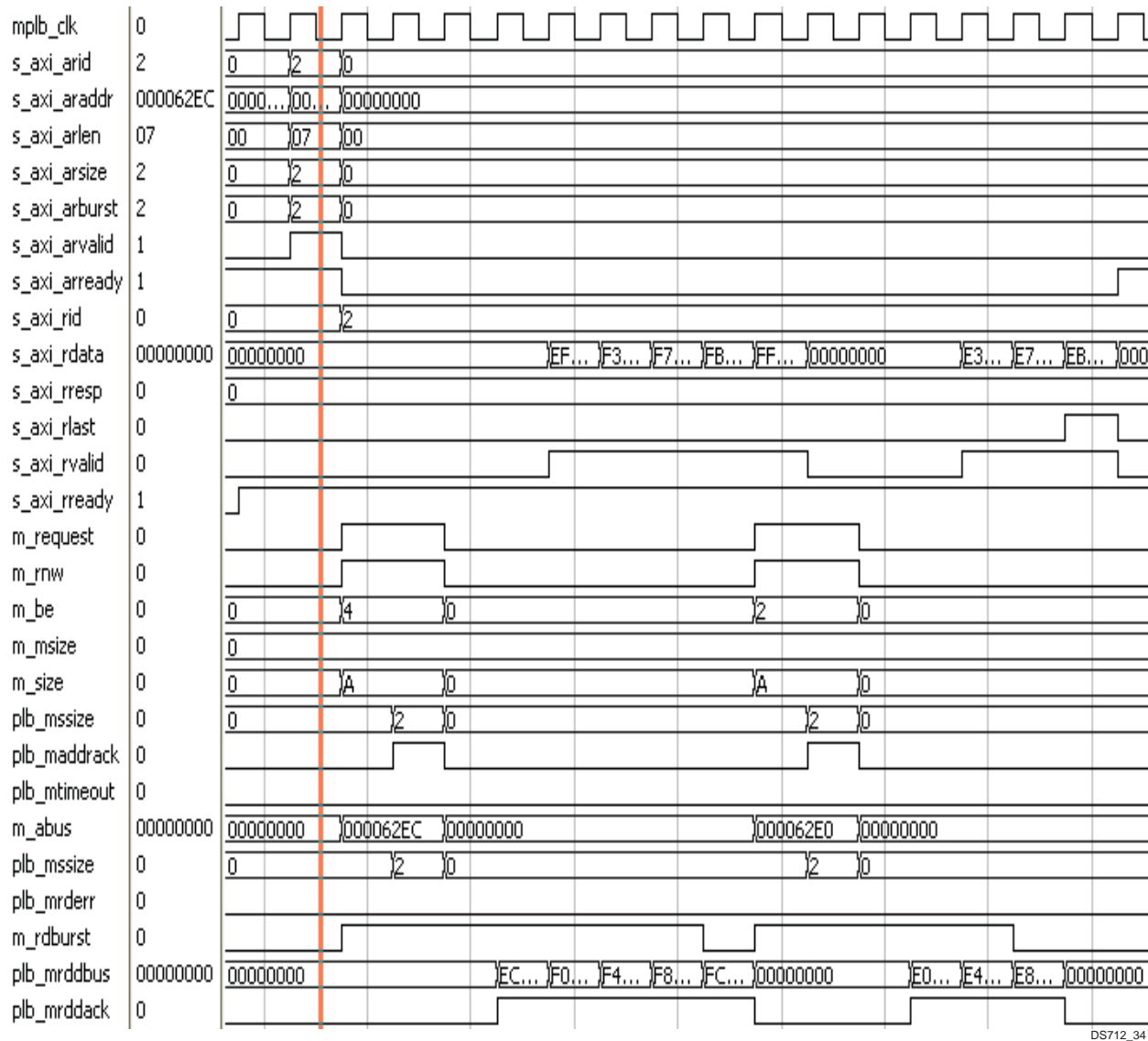
WRAP Read 4 Beat



DS712_33

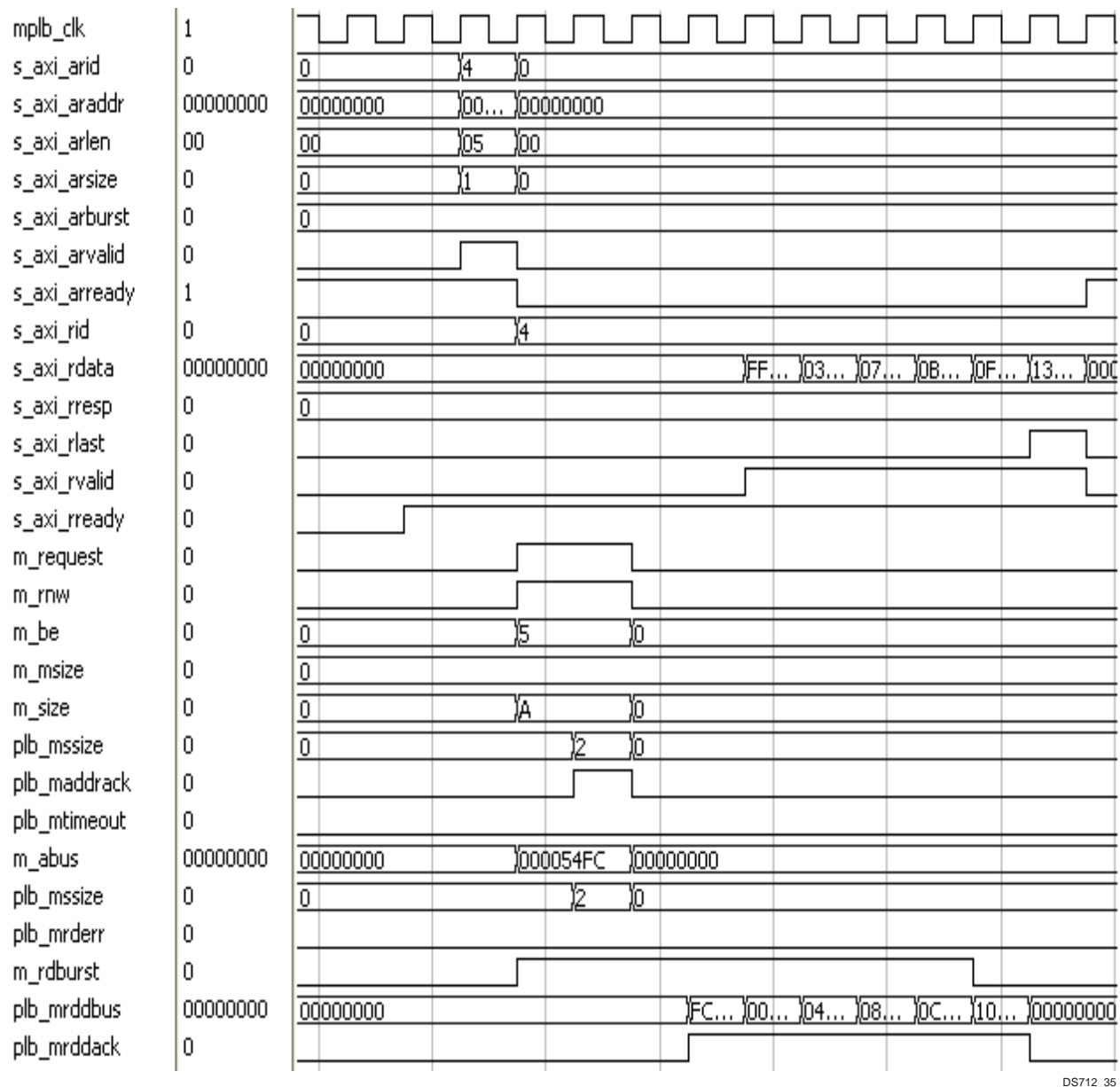
Figure 33: WRAP Read 4 Beat

WRAP Read 8 Beat - 2 xfer



DS712_34

Figure 34: WRAP Read 8 Beat - 2 xfer

FIXED Read 6 Beats

DS712_35

Figure 35: **FIXED Read 6 Beats****Device Utilization and Performance Benchmarks****Core Performance**

Because the AXI to PLBv46 Bridge is a module that will be used with other design pieces in the FPGA, the resource utilization and timing numbers reported in this section are estimates only. When the AXI to PLBv46 Bridge is combined with other pieces of the FPGA design, the utilization of FPGA resources and timing of the design will vary from the results reported here.

The AXI to PLBv46 Bridge resource utilization benchmarks for the Virtex-6 FPGA is shown in [Table 11](#)

Table 11: FPGA Resource Utilization Benchmarks on the Virtex-6 FPGA (XC6VLX130T-FF1156-1)

Parameter Value									Device Resources			Frequency
C_EN_DEBUG_REG	C_S_AXI_SUPPORTS_NARROW_BURST	C_S_AXI_DATA_WIDTH	C_S_AXI_PROTOCOL	C_S_AXI_SUPPORTS_WRITE	C_S_AXI_SUPPORTS_READ	C_S_AXI_WRITE_ACCEPTANCE	C_S_AXI_READ_ACCEPTANCE	C_MPLB_SMALLEST_SLAVE	Slice Registers	Occupied Slices	Slice LUTs	Fmax MHz
NA	NA	32	AXI4LITE	1	1	NA	NA	32	43	119	118	250
NA	0	32	AXI4	0	1	1	1	32	138	183	358	200
0	0	32	AXI4	1	0	1	1	32	137	204	365	200
0	0	32	AXI4	1	1	1	1	32	296	382	721	200
0	1	32	AXI4	1	1	1	1	32	345	457	893	200
0	1	32	AXI4	1	1	2	2	32	462	626	1050	200
1	1	32	AXI4	1	1	1	1	32	330	600	943	200
1	1	32	AXI4	1	1	2	2	32	475	854	1198	200
1	1	64	AXI4	1	1	2	2	32	669	1104	1586	200
1	1	64	AXI4	1	1	2	2	64	604	937	1394	200
0	0	64	AXI4	1	1	1	1	64	373	406	878	200
0	1	64	AXI4	1	1	1	1	64	458	531	1114	200

The AXI to PLBv46 Bridge resource utilization benchmarks for the Spartan-6 FPGA is shown in [Table 12](#)

Table 12: FPGA Resource Utilization Benchmarks on the Spartan-6 FPGA (XC6SLX100T-FGG900-3)

Parameter Value									Device Resources			Frequency
C_EN_DEBUG_REG	C_S_AXI_SUPPORTS_NARROW_BURST	C_S_AXI_DATA_WIDTH	C_S_AXI_PROTOCOL	C_S_AXI_SUPPORTS_WRITE	C_S_AXI_SUPPORTS_READ	C_S_AXI_WRITE_ACCEPTANCE	C_S_AXI_READ_ACCEPTANCE	C_MPLB_SMALLEST_SLAVE	Slices	Slice Registers	LUT	Fmax MHz
NA	NA	32	AXI4LITE	1	1	NA	NA	32	37	117	100	150
NA	0	32	AXI4	0	1	1	1	32	140	185	349	133
0	0	32	AXI4	1	0	1	1	32	156	204	349	133
0	0	32	AXI4	1	1	1	1	32	280	384	713	133
0	1	32	AXI4	1	1	1	1	32	348	461	883	133
0	1	32	AXI4	1	1	2	2	32	415	663	1078	133
1	1	32	AXI4	1	1	1	1	32	385	604	939	133
1	1	32	AXI4	1	1	2	2	32	496	858	1184	120
1	1	64	AXI4	1	1	2	2	32	669	883	1508	115
1	1	64	AXI4	1	1	2	2	64	604	716	1136	110
0	0	64	AXI4	1	1	1	1	64	373	412	865	133
0	1	64	AXI4	1	1	1	1	64	458	533	1120	133

The AXI to PLBv46 Bridge resource utilization benchmarks for the Artix-7 FPGA is shown in [Table 13](#)

Table 13: FPGA Resource Utilization Benchmarks on the Artix-7 FPGA (XC7A350TC-3)

Parameter Value									Device Resources			Frequency
C_EN_DEBUG_REG	C_S_AXI_SUPPORTS_NARROW_BURST	C_S_AXI_DATA_WIDTH	C_S_AXI_PROTOCOL	C_S_AXI_SUPPORTS_WRITE	C_S_AXI_SUPPORTS_READ	C_S_AXI_WRITE_ACCEPTANCE	C_S_AXI_READ_ACCEPTANCE	C_MPLB_SMALLEST_SLAVE	Slices	Slice Registers	LUT	Fmax MHz
NA	NA	32	AXI4LITE	1	1	1	1	32	116	43	117	182
NA	0	32	AXI4	0	1	1	1	32	180	157	338	176
0	0	32	AXI4	1	0	1	1	32	199	154	352	166
0	0	32	AXI4	1	1	1	1	32	374	308	680	162
0	1	32	AXI4	1	1	1	1	32	450	388	879	137
0	1	32	AXI4	1	1	2	2	32	617	526	1,077	157
1	1	32	AXI4	1	1	1	1	32	593	456	929	155
1	1	64	AXI4	1	1	2	2	32	1,094	715	1,618	163
1	1	64	AXI4	1	1	2	2	64	927	630	1,407	161
0	0	64	AXI4	1	1	1	1	64	399	373	885	165
0	1	64	AXI4	1	1	1	1	64	523	478	1,134	166

Read Latency and AXI Bandwidth Utilization

The core is configured for best possible configuration for calculation of latency and bandwidth utilization.

The read latency from address valid (ARVALID) to first data beat (RVALID) (assuming PLB slave latency as 1 clock) of AXI to PLBv46 Bridge is as shown in [Table 14](#). For the latency calculation, it is assumed that PLB slave latency (M_request to PLB_MRdDack) is 1 clock. Latency numbers includes bridge latency and PLB slave latency.

Table 14: Read latency in AXI clocks

C_S_AXI_PROTOCOL	Read Latency
AXI4LITE	2 clocks
AXI4	3 clocks

Best case AXI bandwidth utilization, is calculated on AXI by issuing back to back burst read and write transfers of length 255 and observed in simulation by requesting 1000 transfers, is as shown in Table 15. For improving core performance C_S_AXI_WRITE_ACCEPTANCE and C_S_AXI_READ_ACCEPTANCE need to be set to 2.

Table 15: AXI Bandwidth Utilization

Transfer Type	Utilization in %
Back to back writes	94%
Back to back reads	84%
Back to back reads and writes	83%

Not Supported Features/Limitations

- The bridge does not decode address range for PLB Slave

AXI Master Interface

- AXI data bus width greater than 64 bits
- AXI address bus width is fixed to 32 bits
- AXI Exclusive Accesses
- AXI Trustzone is not supported
- AXI Protection Unit Support is limited
- AXI Low-Power interface is not supported
- In AXI burst write transactions, de-asserted write strobes are supported only in the first and last word of the burst write
- In AXI WRAP write transactions, all the valid byte line must have WSTBs asserted.
- AXI Barrier transactions
- AXI Debug transactions
- AXI user signals
- AXI QOS

PLBv46 Slave Interface

- PLB data bus greater than 64 bits
- PLB address bus width is fixed to 32 bits
- Aborts
- Fixed Length Bursts transfer requests of 17 to 256 data beats
- Fixed Length Bursts of size byte and half word
- Indeterminate Length Bursts
- Premature Fixed Length Burst terminations
- All Cache line transfers
- Transfer attributes
- Pending request and priority input information
- PLBv46 buslock not supported

Reference Documents

The following documents contain reference information important to understanding the AXI to PLBv46 Bridge design:

1. *IBM CoreConnect 128-bit Processor Local Bus: Architecture Specification*, version 4.6
2. *AXI4 AMBA AXI Protocol Version: 2.0 Specification*
3. *Xilinx PLBv46 Interconnect and Interfaces Simplifications and Feature Subset Specification* (Rev 0.6), August 15, 2006
4. DS768, *AXI Interconnect IP Data Sheet*

Support

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

Ordering Information

This Xilinx LogiCORE IP module is provided at no additional cost with the Xilinx ISE Design Suite Embedded Edition software under the terms of the [Xilinx End User License](#). The core is generated using the Xilinx ISE Embedded Edition software (EDK).

Information about this and other Xilinx LogiCORE IP modules is available at the [Xilinx Intellectual Property](#) page. For information on pricing and availability of other Xilinx LogiCORE modules and software, please contact your [local Xilinx sales representative](#).

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
9/21/10	1.0	Initial Xilinx Release.
6/22/10	2.0	Updated core to v2.01a and Xilinx tools to v13.2.

Notice of Disclaimer

The information disclosed to you hereunder (the “Materials”) is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available “AS IS” and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of the Limited Warranties which can be viewed at <http://www.xilinx.com/warranty.htm>; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in Critical Applications: <http://www.xilinx.com/warranty.htm#critapps>.