# LogiCORE IP AXI Master Burst v2.0

## *Product Guide for Vivado Design Suite*

**PG162 December 18, 2013**

XILINX®

XILINX

# Table of Contents

# Introduction

The AXI Master Burst core is a continuation of the Xilinx family of AXI4-compatible LogiCORE™ IP products. It provides a bidirectional interface between a User IP core and the AXI4 interface standard. This version of the AXI Master Burst core has been optimized for bus mastering operations consisting of burst transactions.

# Features

- Compatible with 32-, 64-, and 128-bit AXI4.
- Parameterizable data width of Client IP Interface (IPIC) to 32, 64, or 128 bits.
- Supports AXI4 Read and Write data bursts of 16, 32, 64, 128, and 256 maximum data beats.
  - Transfer width is equal to the parameterized IPIC data width.
  - Automatic AXI4 4K byte address boundary crossing protection.
- User interface consists of a Legacy Command/Status interface and Read and Write LocalLink interfaces for data transactions.

  LocalLink transactions can be up to 1 MB in length (parameterizable) with transfer read data width equal to the LocalLink data width. The AXI Master Burst core automatically breaks up long transaction requests into multiple burst transactions on the AXI4 equal to the parameterized maximum burst length.

| LogiCORE IP Facts Table | |
|---|---|
| **Core Specifics** | |
| Supported Device Family[1] | Zynq®-7000, 7 Series, UltraScale™ Architecture |
| Supported User Interfaces | AXI4 |
| Resources | See Table 2-2, Table 2-3, and Table 2-4 |
| **Provided with Core** | |
| Design Files | VHDL |
| Example Design | Not Provided |
| Test Bench | Not Provided |
| Constraints File | Not Provided |
| Simulation Model | Not Provided |
| Supported S/W Driver | N/A |
| **Tested Design Flows[2]** | |
| Design Entry | Vivado Design Suite |
| Simulation | For supported simulators, see the Xilinx Design Tools: Release Notes Guide. |
| Synthesis | Not Provided. |
| **Support** | |
| Provided by Xilinx @ www.xilinx.com/support | |

**Notes:**

1. For a complete list of supported devices, see Vivado IP catalog.
2. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

# Overview

The AXI Master Burst core is a continuation of the Xilinx family of AXI4-compatible LogiCORE™ IP products. It provides a bidirectional interface between a User IP core and the AXI4 interface standard.

## Feature Summary

- Compatible with 32-, 64-, and 128-bit AXI4.
- Parameterizable data width of Client IP Interface (IPIC) to 32, 64, or 128 bits.
- Supports AXI4 Read and Write data bursts of 16, 32, 64, 128, and 256 maximum data beats.
  - Transfer width is equal to the parameterized IPIC data width.
  - Automatic AXI4 4 KB address boundary crossing protection.
- User interface consisting of a Legacy Command/Status interface and Read and Write LocalLink interfaces for data transactions.

## Applications

The AXI Master Burst core provides a AXI4 mastering capability that has the legacy IPIC User interface suitable for updating to AXI4 those legacy plbv46 designs that used the plbv46_master_burst module.

# Licensing and Ordering Information

This Xilinx LogiCORE IP module is provided at no additional cost with the Xilinx Vivado Design Suite under the terms of the Xilinx End User License.

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 IP modules and tools, contact your local Xilinx sales representative.

# Product Specification

The AXI Master Burst core is designed to provide a quick way to implement a light-weight mastering interface between user logic and AXI4. Figure 2-1 shows a block diagram of the AXI Master Burst core. The port references and groupings are detailed in Table 2-5. The design is parameterizable to transaction data in 32, 64, and 128-bit widths for AXI4 read and write transactions Transaction request protocol between the AXI4 and the User Logic is provided by the IPIC command and Status Adapter Block. The primary data transport function is provided by the Read and Write Controller.

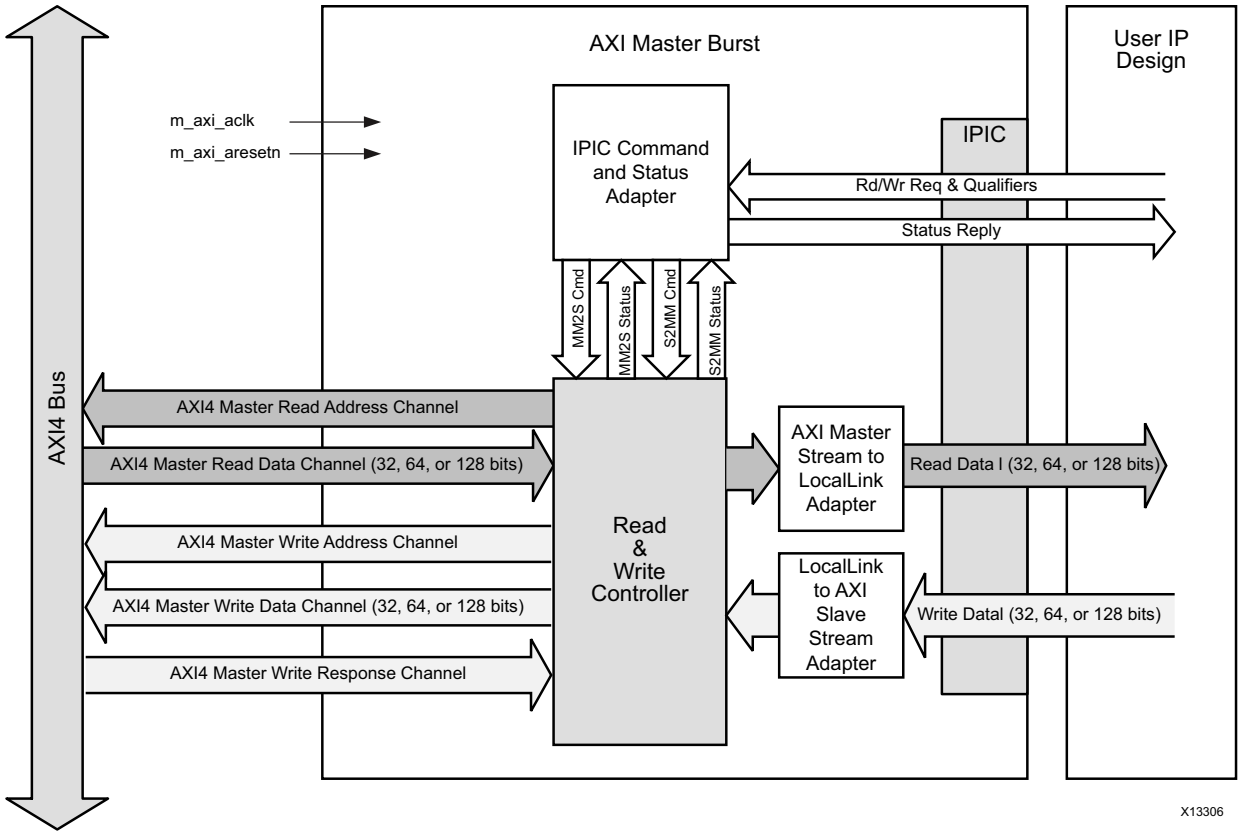


*Figure 2-1:* **AXI Master Burst Block Diagram**

## Typical System Interconnect

The AXI Master Burst core is designed to be instantiated in a User IP design as a helper core. A typical use case is shown in Figure 2. The AXI Master Burst core allows the User IP to access AXI4 slaves via the AXI4 Interconnect.

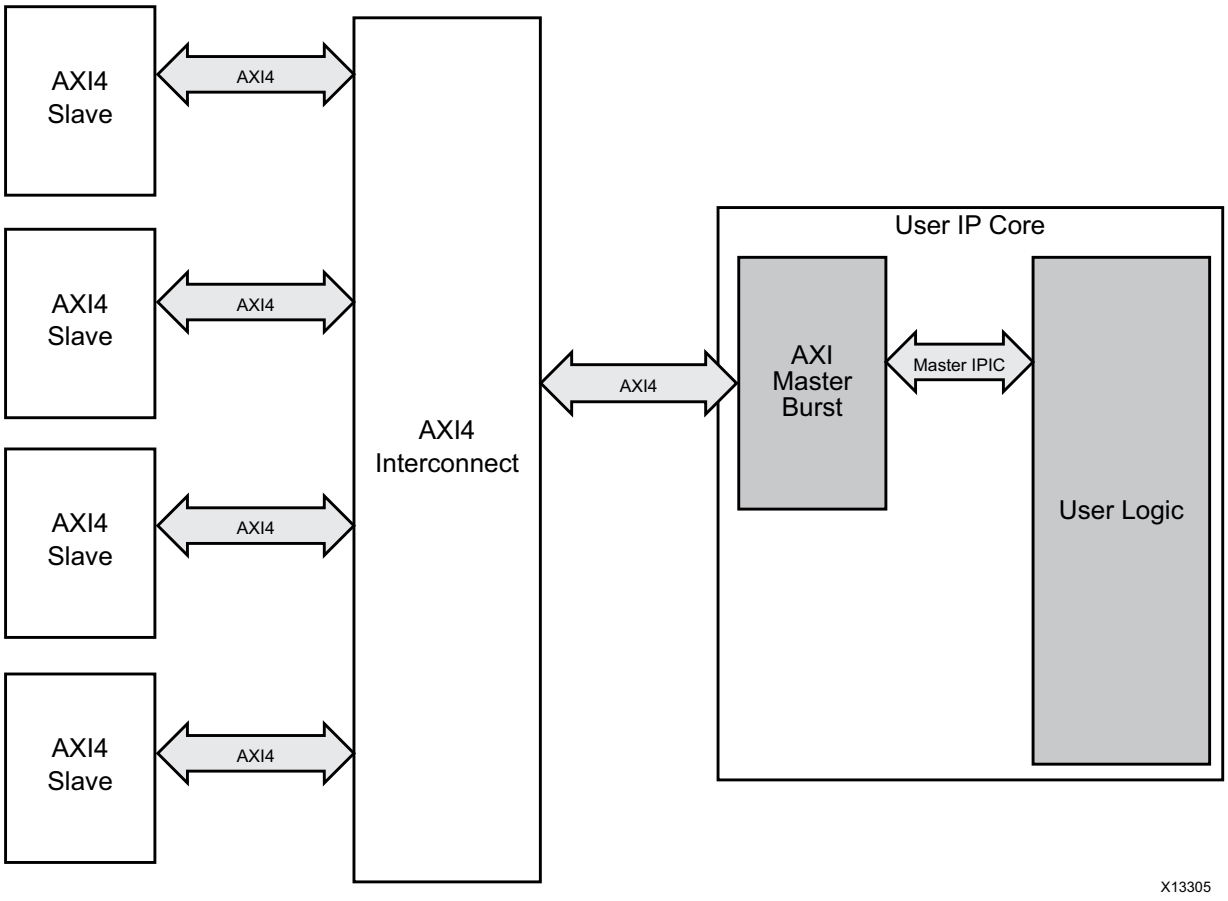


*Figure 2-2:* **Typical System Configuration Using AXI Master Burst**

# Standards

This core supports the AXI4 protocol.

# Performance

This section provides performance information for the AXI Master Burst core.

## Maximum Frequencies

Table 2-1 shows the maximum frequencies for the AXI Master Burst core. Values for the Zynq® family of devices with a Kintex® or Artix® base are expected to be similar to those of the Kintex or Artix family.

*Table 2-1:* **Targeted Frequencies**

| Speed Grade | Family | Fmax (Mhz) |
|---|---|---|
| –1 | Virtex-7 | 180 |
| | Kintex-7 | 180 |
| | Artix-7 | 180 |
| –2 | Virtex-7 | 200 |
| | Kintex-7 | 200 |
| | Artix-7 | 200 |
| –3 | Virtex-7 | 200 |
| | Kintex-7 | 200 |
| | Artix-7 | 200 |

## Latency

There is no information available for this core.

## Throughput

The AXI Master Burst core performs AXI4 burst transactions of 1 to 16, 1 to 32, 1 to 64, 1 to 128, or 1 to 256 data beats per AXI4 request depending on the setting of the C_MAX_BURST_LEN parameter. The AXI Master Burst core breaks up the parent IPIC command (of up to 1,048,575 bytes, see C_LENGTH_WIDTH parameter) into these smaller AXI4 transactions (child commands). In addition, the AXI4 4k address boundary crossing protection is automatically performed by the AXI Master Burst core by ensuring generated transaction requests do not cause a 4k byte address boundary crossing. The core does not add any transaction bubbles between spawned child command data transactions as long as the User logic does not throttle the LocalLink interface and the AXI Interconnect/Target AXI Slave can keep up on the AXI address, AXI data, and in the case of writes, the AXI response channels.

## AXI Address Channel Request Posting Hold-off

The AXI Master Burst core is designed such that it does not begin committing transaction requests in the pertinent AXI Address Channel until the User Logic indicates it is ready via LocalLink ready signaling for at least one clock period after the IPIC input command has been accepted. This is done to optimize the use of the AXI Interconnect and Target AXI Slave by the system. Otherwise, it is possible to stall the AXI Interconnect and Target Slave

with committed transaction requests from the master if the associated data transactions do not occur because the User Logic is not ready to provide write data or accept read data (depending on the type of command).

# Resource Utilization

Resource utilization numbers for the AXI Master Burst core are shown in Table 2-2, Table 2-3, and Table 2-4. UltraScale architecture results are expected to be similar to 7 series device results.

## Artix-7 FPGA

*Table 2-2:* **Performance and Resource Utilization Benchmarks for Artix®-7 FPGA**

| C_NATIVE_DATA_WIDTH | C_MAX_BURST_LEN | C_ADDR_PIPE_DEPTH | C_LENGTH_WIDTH | Slice LUTs | Slice Registers |
|---|---|---|---|---|---|
| 32 | 16 | 4 | 20 | 414 | 611 |

***Note:*** Zynq® devices with an Artix base are expected to have similar utilization.

## Virtex-7 FPGA

*Table 2-3:* **Performance and Resource Utilization Benchmarks for Virtex®-7 FPGA**

| C_NATIVE_DATA_WIDTH | C_MAX_BURST_LEN | C_ADDR_PIPE_DEPTH | C_LENGTH_WIDTH | Slice LUTs | Slice Registers |
|---|---|---|---|---|---|
| 32 | 16 | 4 | 20 | 417 | 611 |

## Kintex-7 FPGA

*Table 2-4:* **Performance and Resource Utilization Benchmarks for Kintex®-7 FPGA**

| C_NATIVE_DATA_WIDTH | C_MAX_BURST_LEN | C_ADDR_PIPE_DEPTH | C_LENGTH_WIDTH | Slice LUTs | Slice Registers |
|---|---|---|---|---|---|
| 32 | 16 | 4 | 20 | 416 | 611 |

***Note:*** Zynq devices with a Kintex base are expected to have similar utilization.

# Port Descriptions

This section provides port information for the AXI Master Burst core.

## I/O Signals

The AXI Master Burst signals are described in Table 2-5.

*Table 2-5:* **AXI Master Burst I/O Signal Description**

| Signal Name | Interface | Signal Type | Init Status | Description |
|---|---|---|---|---|
| **System Signals** | | | | |
| m_axi_aclk | Clock | I | | **AXI Master Burst Clock.** |

*Table 2-5:* **AXI Master Burst I/O Signal Description *(Cont'd)***

| Signal Name | Interface | Signal Type | Init Status | Description |
|---|---|---|---|---|
| m_axi_aresetn | Reset | I | | **AXI Master Burst Reset.** When asserted low, the AXI Master Burst core is put into hard reset. This signal must be synchronous to m_axi_aclk. |
| **Master Detected Error Discrete** | | | | |
| md_error | Discrete Out | O | | **AXI Master Burst Master Detected Error.** Active high master detected error output discrete. This bit is sticky when set and is only cleared by a hardware reset. |
| **AXI4 Master Read Address Channel** | | | | |
| m_axi_arready | m_axi | I | | **AXI Master Burst Read Address Channel Read Address Ready.** Indicates target is ready to accept the read address.<br>• 1 = Target read to accept address.<br>• 0 = Target not ready to accept address. |
| m_axi_arvalid | m_axi | O | 0 | **AXI Master Burst Read Address Channel Read Address Valid.** Indicates if m_axi_araddr is valid.<br>• 1 = Read Address is valid.<br>• 0 = Read Address is not valid. |
| m_axi_araddr(C_M_AXI_ADDR_WIDTH-1: 0) | m_axi | O | zeros | **AXI Master Burst Read Address Channel Address Bus.** The starting address for the requested read transaction. |
| m_axi_arlen(7:0) | m_axi | O | zeros | **AXI Master Burst Read Address Channel Burst Length.** This qualifier specifies the requested AXI Read transaction length In data beats - 1. |
| m_axi_arsize(2:0) | m_axi | O | zeros | **AXI Master Burst Read Address Channel Burst Size.** Indicates the data transaction width of each burst data beat.<br>• 000b = Not supported.<br>• 001b =Not supported.<br>• 010b = 4 bytes (32-bit wide burst).<br>• 011b = 8 bytes (64-bit wide burst).<br>• 100b = 16 bytes (128-bit wide burst).<br>• 101b = Not supported.<br>• 110b = Not supported.<br>• 111b = Not supported. |
| m_axi_arburst(1:0) | m_axi | O | zeros | **AXI Master Burst Read Address Channel Burst Type.** Indicates type of burst.<br>• 00b = FIXED - Not supported.<br>• 01b = INCR - Incrementing address.<br>• 10b = WRAP - Not supported.<br>• 11b = Reserved. |

*Table 2-5:* **AXI Master Burst I/O Signal Description *(Cont'd)***

| Signal Name | Interface | Signal Type | Init Status | Description |
|---|---|---|---|---|
| m_axi_arprot(2:0) | m_axi | O | 000b | **AXI Master Burst Read Address Channel Protection.** This is always driven with a constant output of 000b. |
| m_axi_arcache(3:0) | m_axi | O | 0011b | **AXI Master Burst Read Address Channel Cache.** This is always driven with a constant output of 0011b. |
| **AXI4 Master Read Data Channel** | | | | |
| m_axi_rready | m_axi | O | 0 | **AXI Master Burst Read Data Channel Ready.** Indicates the read channel is ready to accept read data.<br>• 1 = Is ready.<br>• 0 = Is not ready. |
| m_axi_rvalid | m_axi | I | | **AXI Master Burst Read Data Channel Data Valid.** Indicates m_axi_rdata is valid.<br>• 1 = Valid read data.<br>• 0 = Not valid read data. |
| m_axi_rdata(C_M_AXI_DATA_WIDTH-1: 0) | m_axi | I | | **AXI Master Burst Read Data Channel Read Data**. Read data bus for the requested read transaction. |
| m_axi_rresp(1:0) | m_axi | I | | **AXI Master Burst Read Data Channel Response.** Indicates results of the read transaction.<br>• 00b = OKAY - Normal access has been successful.<br>• 01b = EXOKAY - Not supported.<br>• 10b = SLVERR - Slave returned error on transaction.<br>• 11b = DECERR - Decode error, transaction targeted unmapped address. |
| m_axi_rlast | m_axi | I | | **AXI Master Burst Read Data Channel Last.** Indicates the last data beat of a burst transaction.<br>• 0 = Not last data beat.<br>• 1 = Last data beat. |
| **AXI4 Master Write Address Channel** | | | | |
| m_axi_awready | m_axi | I | | **AXI Master Burst Write Address Channel Write Address Ready.** Indicates target is ready to accept the write address.<br>• 1 = Target ready to accept address.<br>• 0 = Target not ready to accept address. |
| m_axi_awvalid | m_axi | O | 0 | **AXI Master Burst Write Address Channel Write Address Valid.** Indicates if m_axi_awaddr is valid.<br>• 1 = Write Address is valid.<br>• 0 = Write Address is not valid. |

*Table 2-5:* **AXI Master Burst I/O Signal Description *(Cont'd)***

| Signal Name | Interface | Signal Type | Init Status | Description |
|---|---|---|---|---|
| m_axi_awaddr (C_M_AXI_ADDR_ WIDTH-1: 0) | m_axi | O | zeros | **AXI Master Burst Write Address Channel Address Bus**. The starting address for the requested write transaction. |
| m_axi_awlen(7:0) | m_axi | O | zeros | **AXI Master Burst Write Address Channel Burst Length.** This qualifier specifies the requested AXI Write transaction length In data beats - 1. |
| m_axi_awsize(2:0) | m_axi | O | zeros | **AXI Master Burst Write Address Channel Burst Size.** Indicates the data transaction width of each burst data beat.<br>• 000b = Not supported.<br>• 001b =Not supported.<br>• 010b = 4 bytes (32-bit wide burst).<br>• 011b = 8 bytes (64-bit wide burst).<br>• 100b = 16 bytes (128-bit wide burst).<br>• 101b = Not supported.<br>• 110b = Not supported.<br>• 111b = Not supported. |
| m_axi_awburst(1:0) | m_axi | O | zeros | **AXI Master Burst Write Address Channel Burst Type.** Indicates type of burst.<br>• 00b = FIXED - Not supported.<br>• 01b = INCR - Incrementing address.<br>• 10b = WRAP - Not supported.<br>• 11b = Reserved. |
| m_axi_awprot(2:0) | m_axi | O | 000b | **AXI Master Burst Write Address Channel Protection.** This is always driven with a constant output of 000b. |
| m_axi_awcache(3:0) | m_axi | O | 0011b | **AXI Master Burst Write Address Channel Cache.** This is always driven with a constant output of 0011b. |
| **AXI4 Master Write Data Channel** | | | | |
| m_axi_wready | m_axi | I | | **AXI Master Burst Write Data Channel Ready.** Indicates the SG Write Data Channel target slave is ready to accept<br>write data.<br>• 1 = Target slave is ready.<br>• 0 = Target slave is not ready. |
| m_axi_wvalid | m_axi | O | 0 | **AXI Master Burst Write Data Channel Data Valid.** Indicates the Write Data Channel has a valid data beat on the bus.<br>• 1 = Valid write data.<br>• 0 = Not valid write data. |
| m_axi_wdata (C_M_AXI_DATA_ WIDTH-1: 0) | m_axi | O | zeros | **AXI Master Burst Write Data Channel Write Data Bus.** |

*Table 2-5:* **AXI Master Burst I/O Signal Description *(Cont'd)***

| Signal Name | Interface | Signal Type | Init Status | Description |
|---|---|---|---|---|
| m_axi_wstrb (C_M_AXI_DATA_ WIDTH/8 - 1: 0) | m_axi | O | zeros | **AXI Master Burst Write Data Channel Write Strobe Bus.** |
| m_axi_wlast | m_axi | O | | **AXI Master Burst Write Data Channel Last.** Indicates the last data beat of a burst transaction.<br>• 1 = Last data beat.<br>• 0 = Not last data beat. |
| **AXI4 Master Write Response Channel** | | | | |
| m_axi_bresp(1:0) | m_axi | I | | **AXI Master Burst Write Response Channel Response.** Indicates results of the write transaction.<br>• 00b = OKAY - Normal access has been successful.<br>• 01b = EXOKAY - Not supported.<br>• 10b = SLVERR - Slave returned error on transaction.<br>• 11b = DECERR - Decode error, transaction targeted unmapped address. |
| m_axi_bvalid | m_axi | I | | **AXI Master Burst Write Response Channel Response Valid.** Indicates response, m_axi_bresp, is valid.<br>• 1 = Response is valid.<br>• 0 = Response is not valid. |
| m_axi_bready | m_axi | O | 0 | **AXI Master Burst Write Response Channel Ready**. Indicates source is ready to receive response.<br>• 1 = Ready to receive response.<br>• 0 = Not ready to receive response. |
| **IPIC Command Interface Signals** | | | | |
| ip2bus_mstrd_req | IPIC | I | | **AXI Master Burst Read Request.** Active high read request initiation control signal |
| ip2bus_mstwr_req | IPIC | I | | **AXI Master Burst Write Request.** Active high write request initiation control signal |
| ip2bus_mst_addr(C_M_AXI _ADDR_WIDTH -1:0) | IPIC | I | | **AXI Master Burst Address.** Address to be used for the specified read or write command |
| ip2bus_mst_be((C_NATIVE _DATA_WIDTH/8) -1: 0) | IPIC | I | | **AXI Master Burst Byte Enables.** Input command qualifiers (active high byte enables) used to indicate the valid bytes for the specified read or write transaction. This input is only used for requests when the IP2Bus_Mst_Type qualifier is set to '0' for a single data beat.<br>If ip2bus_mst_be is set to all zeros for a single beat request, an internal error is induced in the AXI Master Burst core. |

*Table 2-5:* **AXI Master Burst I/O Signal Description *(Cont'd)***

| Signal Name | Interface | Signal Type | Init Status | Description |
|---|---|---|---|---|
| ip2bus_mst_length(C_LENGTH_WIDTH-1:0) | IPIC | I | | **AXI Master Burst Transaction Length.** Input command qualifier specifying the length of the requested transaction in bytes. The max value that can be specified is $2^n$-1where n is the value assigned to the C_LENGTH_WIDTH parameter. This is only used for Burst type transactions and a value of zero is not allowed.<br>If the ip2bus_mst_length is set to all zeros for a burst request, an internal error is induced in the AXI Master Burst core. |
| ip2bus_mst_type | IPIC | I | | **AXI Master Burst Transaction Type.** Input command qualifier specifying the type of transaction being requested by the User logic.<br>• 0 = Single Data Beat<br>• 1 = Fixed Length Burst |
| ip2bus_mst_lock | IPIC | I | | **AXI Master Burst Lock.** This input command qualifier is ignored by the AXI Master Burst core. |
| ip2bus_mst_reset | IPIC | I | | **AXI Master Burst Reset.** Active high reset input used to reset all of the AXI Master Burst logic.<br>This input should not be asserted after a transaction command has been posted to the Command Interface and before the transaction completion (as indicated by the assertion of the Bus2IP_Mst_Cmplt output status signal) has occurred. |
| bus2ip_mst_cmdack | IPIC | O | '0' | **AXI Master Burst Command Acknowledge.** Active high signal indicating that the Command Request (Read or Write) has been posted to and accepted by the Read/Write Controller. |
| bus2ip_mst_cmplt | IPIC | O | '0' | **AXI Master Burst Command Complete.** Active high signal indicating the requested transaction has completed by the Read/Write Controller and the associated status bits are valid to sample. |
| bus2ip_mst_error | IPIC | O | '0' | **AXI Master Burst Error.** Active high signal indicating an error was encountered by the Read/Write Controller during the requested transaction. This signal is valid when Bus2IP_Mst_Cmplt is asserted. |
| bus2ip_mst_rearbitrate | IPIC | O | '0' | **AXI Master Burst Rearbitrate.** Not part of AXI4. This signal is always set to '0'. |
| bus2ip_mst_timeout | IPIC | O | '0' | **AXI Master Burst Timeout.** Not part of AXI4. This signal is always set to '0'. |
| **IPIC Read LocalLink Data Interface Signals (Note: Legacy IPIC is Big Endian)** | | | | |
| bus2ip_mstrd_d(C_NATIVE_DATA_WIDTH - 1:0) | IPIC | O | zeros | **AXI Master Burst Read LocalLink Data**. The Read LocalLink data output bus. |

*Table 2-5:* **AXI Master Burst I/O Signal Description *(Cont'd)***

| Signal Name | Interface | Signal Type | Init Status | Description |
|---|---|---|---|---|
| bus2ip_mstrd_rem((C_NATIVE_DATA_WIDTH/8])- 1:0) | IPIC | O | ones | **AXI Master Burst Read LocalLink Remainder**. REM output presented in active low mask format. |
| bus2ip_mstrd_sof_n | IPIC | O | '1' | **AXI Master Burst Read LocalLink Start of Frame**. Active low signal indicating the starting data beat of a Read LocalLink packet transaction. |
| bus2ip_mstrd_eof_n | IPIC | O | '1' | **AXI Master Burst Read LocalLink End of Frame**. Active low signal indicating the ending data beat of a Read LocalLink packet transaction. |
| bus2ip_mstrd_src_rdy_n | IPIC | O | '1' | **AXI Master Burst Read LocalLink Source Ready**. Active low signal indicating that the data value asserted on the Bus2IP_MstRd_d output bus is valid and ready for transaction. |
| bus2ip_mstrd_src_dsc_n | IPIC | O | '1' | **AXI Master Burst Read LocalLink Source Discontinue**. Active low signal indicating that the Read LocalLink Source (master) needs to discontinue the transaction. This is only asserted by the AXI Master Burst core if an internal error is encountered by the MM2S side of the Read/Write Controller. The bus2ip_mstrd_src_dsc_n is asserted (in conjunction with bus2ip_mstrd_eof_n) until the LocalLink data beat is completed with acceptance by the destination via assertion of ip2bus_mstrd_dst_rdy_n. |
| ip2bus_mstrd_dst_rdy_n | IPIC | I | | **AXI Master Burst Read LocalLink Destination Ready**. Active low input signal indicating that the LocalLink destination (User logic) is ready for a data transaction beat.<br>The AXI Master Burst core does not issue any read requests on the AXI Read Address Channel until the ip2bus_mstrd_dst_rdy_n is asserted low for at least 1 clock period after the IPIC read command has been issued. |
| ip2bus_mstrd_dst_dsc_n | IPIC | I | | **AXI Master Burst Read LocalLink Destination Discontinue**. Active low signal indicating that the Read LocalLink Destination (User logic) needs to discontinue the transaction. This is not supported by the AXI Master Burst core. User logic should tie this signal to a constant logic high. |
| **IPIC Write Data Interface Signals (Note: Legacy IPIC is Big Endian)** | | | | |
| ip2bus_mstwr_d(C_NATIVE_DATA_WIDTH - 1:0) | IPIC | I | | **AXI Master Burst Write LocalLink Data.** Write Data input |
| ip2bus_mstwr_rem([C_NATIVE_DATA_WIDTH/8] - 1:0) | IPIC | I | | **AXI Master Burst Write LocalLink Remainder**. REM input presented in active low mask format. |

*Table 2-5:* **AXI Master Burst I/O Signal Description *(Cont'd)***

| Signal Name | Interface | Signal Type | Init Status | Description |
|---|---|---|---|---|
| ip2bus_mstwr_src_rdy_n | IPIC | I | | **AXI Master Burst Write LocalLink Source Ready**. Active low input signal indicating that the data and qualifiers asserted on the Write LocalLink bus are valid (from the source) and ready for a transaction data beat.<br>The AXI Master Burst core does not issue any write requests on the AXI Write Address Channel until the ip2bus_mstwr_src_rdy_n is asserted low for at least 1 clock period after the IPIC write command has been issued. |
| ip2bus_mstwr_src_dsc_n | IPIC | I | | **AXI Master Burst Write LocalLink Source Discontinue**. Active low input signal indicating that the Write LocalLink Source (User Logic) needs to discontinue the transaction. This is not supported by the AXI Master Burst core and should be tied to logic '1' by the User logic. |
| ip2bus_mstwr_sof_n | IPIC | I | | **AXI Master Burst Write LocalLink Start of Frame**. Active low input signal indicating the starting data beat of a Write LocalLink packet transaction. |
| ip2bus_mstwr_eof_n | IPIC | I | | **AXI Master Burst Write LocalLink End of Frame**. Active low input signal indicating the ending data beat of a Write LocalLink packet transaction. |
| bus2ip_mstwr_dst_rdy_n | IPIC | O | '1' | **AXI Master Burst Write LocalLink Destination Ready.** Active low output signal indicating that the AXI Master Burst core is ready to accept a data transaction on the Write LocalLink. |
| bus2ip_mstwr_dst_dsc_n | IPIC | O | '1' | **AXI Master Burst Write LocalLink Destination Discontinue.** Active low output signal indicating that the AXI Master Burst core needs to discontinue Write LocalLink operations. This is only asserted by the AXI Master Burst core if an internal error is encountered by the S2MM side of the Read/Write Controller. The bus2ip_mstwr_dst_dsc_n is asserted until the LocalLink data beat is completed. The bus2ip_mstwr_dst_dsc_n is asserted until the LocalLink data beat is completed. The Master Burst core still expects the Write LocalLink to be terminated with a ip2bus_mstwr_eof_n assertion in the next LocalLink data beat. |

# Designing with the Core

This chapter includes guidelines and additional information to facilitate designing with the AXI Master Burst core.

## General Design Guidelines

The AXI Master Burst design parameters are listed and described in Table 3-1.

*Table 3-1:* **AXI Master Burst Design Parameter Description**

| Parameter Name | Allowable Values | Default Values | VHDL Type | Feature/Description |
|---|---|---|---|---|
| **AXI Master Burst General Parameters** | | | | |
| C_FAMILY | virtex7, kintex7, artix7, zynq7000, ultrascale | virtex7 | String | Specifies the target device family. |
| **AXI Master Burst AXI4 Parameters** | | | | |
| C_M_AXI_ADDR_WIDTH | 32 | 32 | integer | Address width (in bits) of AXI4 Interface. This is currently fixed at 32 bits. |
| C_M_AXI_DATA_WIDTH | 32, 64, 128, 256 | 32 | integer | Data width (in bits) of AXI4 Interface. |
| C_MAX_BURST_LEN | 16, 32, 64, 128, 256 | 16 | integer | Specifies the maximum number of data beats to use for each AXI transaction initiated by the AXI Master Burst core. |
| C_ADDR_PIPE_DEPTH | 1-14 | 1 | integer | Specifies the depth of the address pipelining the AXI Master Burst core will support when submitting transaction requests to the AXI Address Channels. |

*Table 3-1:* **AXI Master Burst Design Parameter Description *(Cont'd)***

| Parameter Name | Allowable Values | Default Values | VHDL Type | Feature/Description |
|---|---|---|---|---|
| C_NATIVE_DATA_WIDTH | 32, 64, 128 | 32 | integer | Data width (in bits) of LocalLink Data Interface. The value assigned must be less than or equal to the value assigned to the C_M_AXI_DATA_WIDTH parameter |
| C_LENGTH_WIDTH | 12-20 | 12 | integer | Specifies the width (in bits) of the IPIC ip2bus_mst_length input port. The value limits the maximum number of bytes to be transacted that can be specified by the user on the ip2bus_mst_length input port. |

## Parameter Descriptions

### C_FAMILY

- **Type:** string
- **Allowed Values:** 7 Series, Zynq-7000®, UltraScale™ devices
- **Definition:** Indicates the target device family for the design
- **Description:** This parameter is set by the Vivado® tools to a value reflecting the device family selected for the Vivado project.

### C_M_AXI_ADDR_WIDTH

- **Type:** Integer
- **Allowed Values:** 32 (default = 32)
- **Definition:** Bit width of the AXI Read and AXI Write Address Channels on the AXI Master Burst AXI4 interface.
- **Description:** This integer parameter is used to size the Read Address and Write Address Channels of the AXI4 Master Burst interface. The Vivado tool suite assigns this parameter a fixed value of 32.

### C_M_AXI_DATA_WIDTH

- **Type:** Integer
- **Allowed Values:** 32, 64, 128, or 256 (default = 32)
- **Definition:** Bit width of the AXI Read and AXI Write Data Channels on the AXI Master Burst AXI4 interface

- **Description:** This integer parameter is used to size the Read Data and Write Data Channels of the AXI4 AXI Master Burst interface.

### C_MAX_BURST_LEN

- **Type:** Integer
- **Allowed Values:** 16, 32, 64, 128, or 256 (default = 16)
- **Definition:** This parameter limits the burst length requested by the AXI Master Burst core on the AXI4 data transport interface.

### C_NATIVE_DATA_WIDTH

- **Type:** Integer
- **Allowed Values:** 32, 64, or 128 (default = 32)
- **Definition:** Defines the bit width of the LocalLink Read and Write data channels
- **Description:** This integer parameter is used to size the Read LocalLink Data and Write LocalLink Data Channels of the AXI4 AXI Master Burst IPIC interface. The value assigned must be less than or equal to the value assigned to the C_M_AXI_DATA_WIDTH parameter.

### C_ADDR_PIPE_DEPTH

- **Type:** Integer
- **Allowed Values:** 1-14 (default = 1)
- **Definition:** Sets the address pipeline limit used by the AXI Master Burst core for posting requests on the AXI Address Channels
- **Description:** The effective address pipelining on the AXI4 Read and Write Address Channels will be the value assigned plus 2. If the value assigned is 1, the effective address pipelining will be 2.

### C_LENGTH_WIDTH

- **Type:** Integer
- **Allowed Values:** 12 to 20 (default = 12)
- **Definition:** Sets the bit width of the IPIC ip2bus_mst_length command qualifier.
- **Description:** The bit width allows a maximum of $2^n$-1 bytes to be specified for transaction per command submitted by the User on the IPIC Command interface.
  - 12 bits = 4,095 bytes max per command
  - 13 bits = 8,191 bytes max per command

- 14 bits = 16,383 bytes max per command
- 15 bits = 32,767 bytes max per command
- 16 bits = 65,535 bytes max per command
- 17 bits = 131,071 bytes max per command
- 18 bits = 262,143 bytes max per command
- 19 bits = 524,287 bytes max per command
- 20 bits = 1,048,575 bytes max per command

### Parameter - I/O Signal Dependencies

*Table 3-2:* **Parameter - I/O Signal Dependencies**

| Parameter Name | Affects Port | Depends on Parameter | Relationship Description |
|---|---|---|---|
| C_M_AXI_DATA_WIDTH | m_axi_rdata<br>m_axi_wdata<br>m_axi_wstrb | | The value assigned to the parameter sets the vector width of the affected port. |
| C_M_AXI_ADDR_WIDTH | ip2bus_mst_addr<br>m_axi_awaddr<br>m_axi_araddr | | The value assigned to the parameter sets the vector width of the affected port. |
| C_NATIVE_DATA_WIDTH | ip2bus_mst_be<br>ip2bus_mstwr_d<br>bus2ip_mstrd_d<br>ip2bus_mstwr_rem<br>bus2ip_mstrd_rem | | The value assigned to the parameter sets the vector width of the affected port. |
| C_NATIVE_DATA_WIDTH | | C_M_AXI_DATA_WIDTH | The value assigned to C_NATIVE_DATA_WIDTH must be less than or equal to the value assigned to C_M_AXI_DATA_WIDTH |
| C_LENGTH_WIDTH | ip2bus_mst_length | | The value assigned to the parameter sets the vector width of the affected port. |

# Clocking

The AXI Master Burst core utilizes a single clock for logic synchronization. This clock is input on the `m_axi_aclk` input port. All interfaces for the core are required to be synchronized to this clock. The AXI Master Burst core has been simulation tested with an `m_axi_aclk` frequency range of 10 MHz to 200 MHz. Actual Fmax achieved in a hardware implementation can vary. See the section Performance.

# Resets

An active low reset assertion on the AXI Master Burst `m_axi_aresetn` input resets the entire AXI Master Burst core. This is considered a hardware reset and there are no graceful completions of AXI4 transactions in progress. A hardware reset initializes all AXI Master Burst internal logic to power on conditions. It is required that the `m_axi_aresetn` input is synchronous to the `m_axi_aclk` master clock input and is asserted for the minimum number of clocks stated in Table 3-3. The table also indicates the stabilization time for AXI Master Burst outputs reacting to a reset condition.

*Table 3-3:* **Reset Assertion/Deassertion Stabilization Times**

| Description | Value | Applicable Signal |
|---|---|---|
| Minimum assertion time | 8 clocks (m_axi_aclk) | axi_resetn input |
| Reset assertion to output signals in reset state (maximum) | 3 clocks (m_axi_aclk) | All output signals |
| Reset deassertion to normal operation state (maximum) | 3 clocks (m_axi_aclk) | All output signals |

The input signal `Bus2IP_Mst_Reset` should not be asserted after a transaction command has been posted to the AXI Master Burst command interface and before it has completed with the assertion of the `Bus2IP_Mst_Cmplt`. To do so can cause the master to violate the AXI4 protocol and hang the AXI4 system connected to the master.

# Protocol Description

Refer to the *ARM® AMBA® AXI Protocol v2.0* [Ref 1].

## Transaction Timing Examples

The next section shows timing relationships for AXI4 and the IPIC interface signals during read and write transactions. Actual timing relationships can vary depending on AXI handshaking and LocalLink handshaking.

### Single Data Beat Read Operation

A single beat read cycle is shown in Figure 3-1. The diagram shows the AXI Slave accepting the read address and qualifiers in one clock cycle and presenting the read data in the next clock cycle.

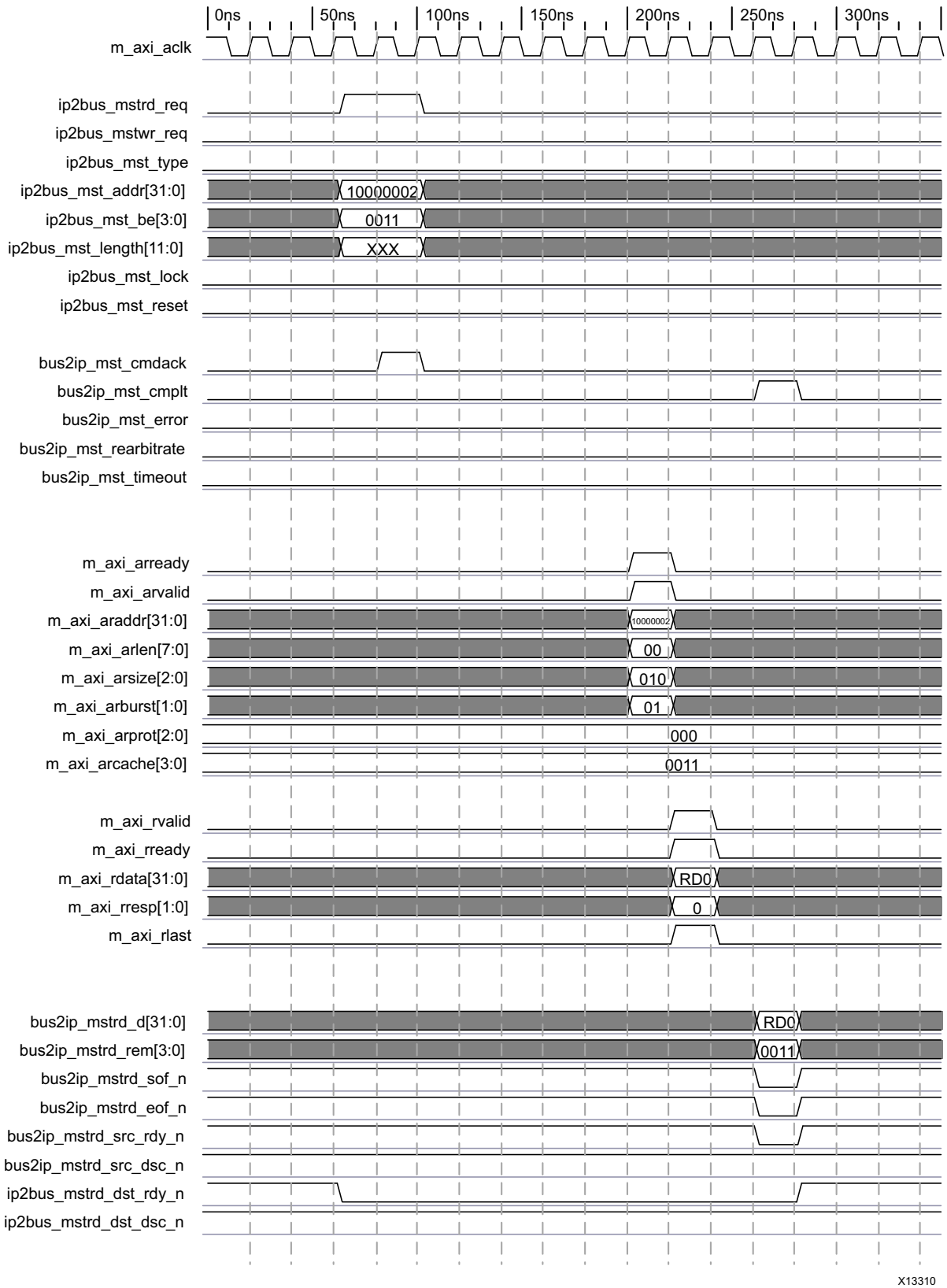


*Figure 3-1:* **Example Single Beat Read Transaction Timing**

## Single Data Beat Write Operation

A typical single beat write cycle is shown in Figure 3-2.

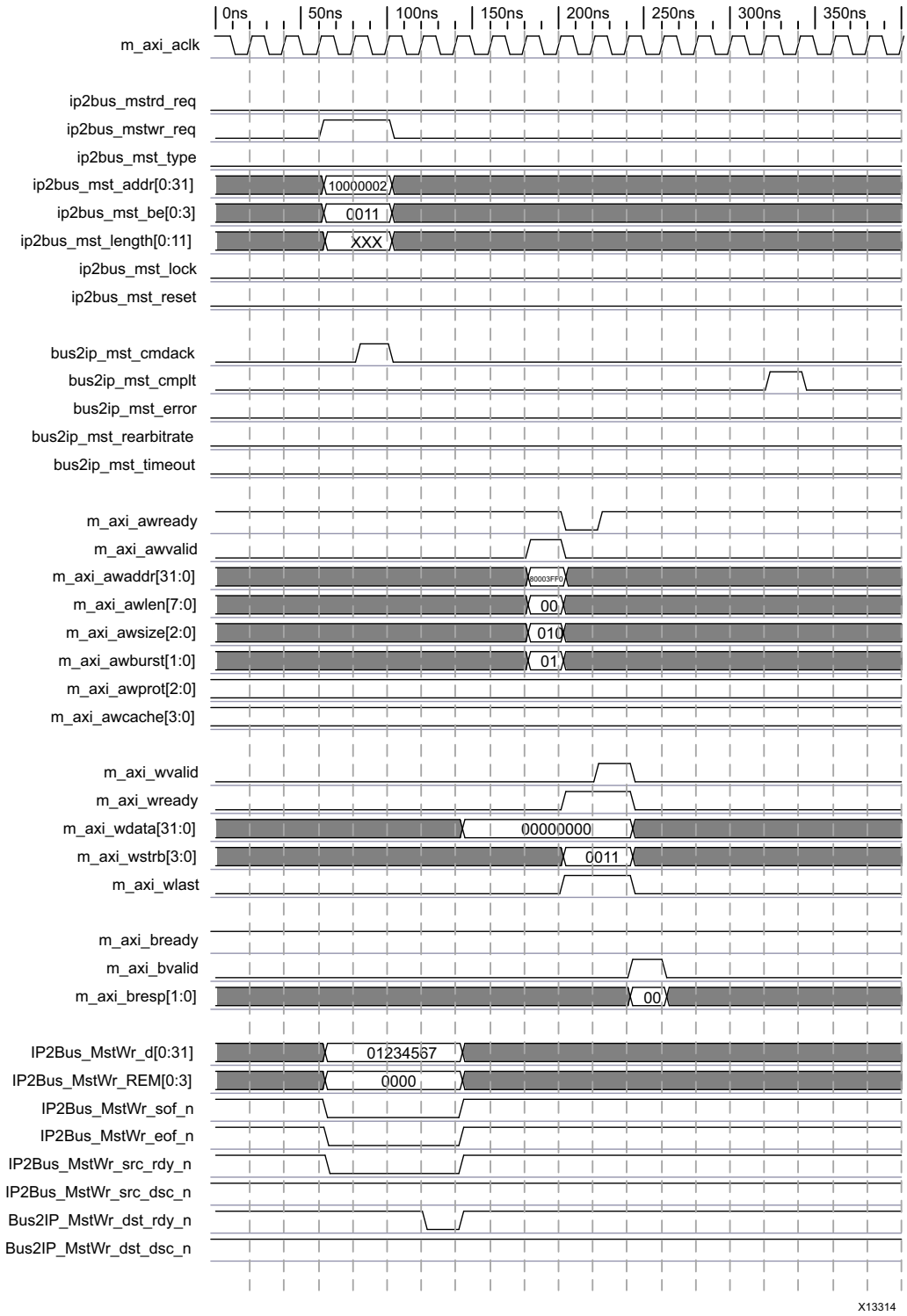


*Figure 3-2:* **Example Single Beat Write Transaction Timing**

## Single Data Beat Read Operation with AXI Read Data Channel Reported Error

A single data beat Read transaction with a Slave reported error is shown in Figure 3-3. The AXI Read Data Channel response error is captured, reported on the IPIC Status Channel, and the master's `md_error` output is asserted and held. The assertion of `md_error` is cleared by the `ip2bus_mst_reset` input from the IPIC Command interface. The `m_axi_aresetn` would clear the `md_error` if it were asserted.

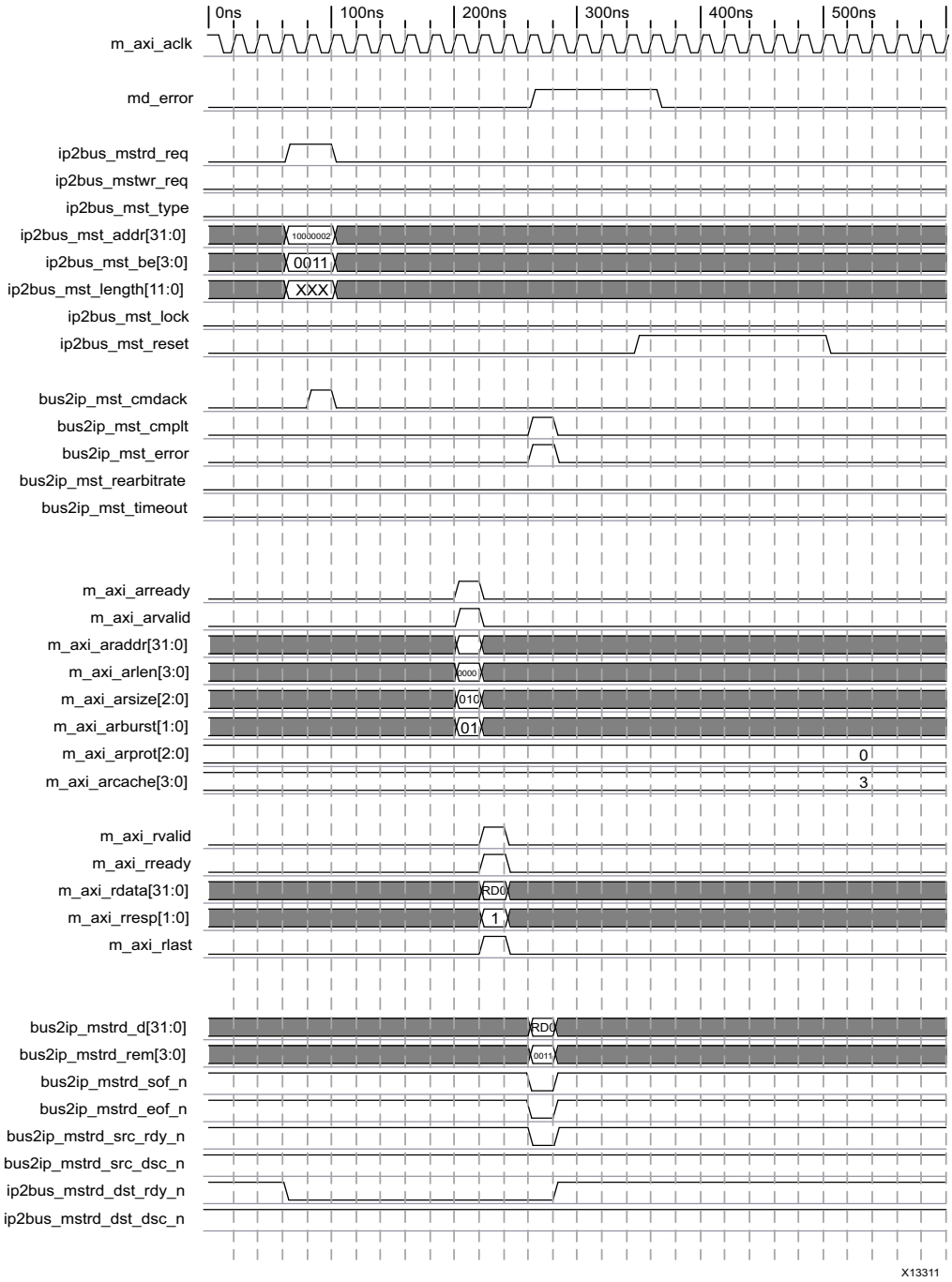


*Figure 3-3:* **Example Single Beat Read Transaction Timing With Error**

## Single Data Beat Write Operation with AXI Response Channel Reported Error

A single beat write transaction is shown in Figure 3-4. The AXI Write Response Channel response error is captured, reported on the IPIC Status Channel, and the master's `md_error` output is asserted and held. The assertion of `md_error` is cleared by the `ip2bus_mst_reset` input from the IPIC Command interface. The `m_axi_aresetn` would clear the `md_error` if it were asserted.

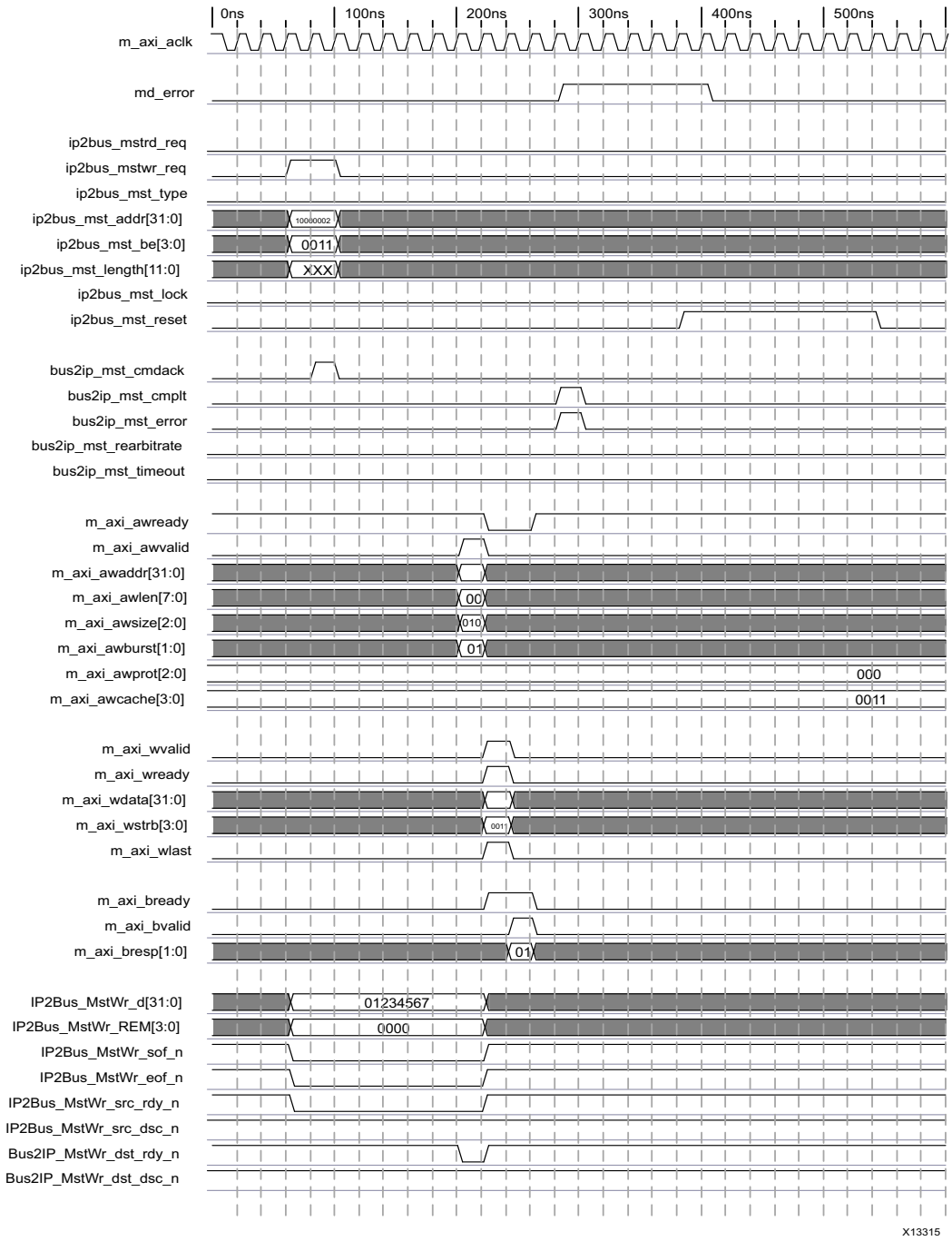


*Figure 3-4:* **Example Single Beat Write Transaction Timing With Error**

## Burst Read Transaction

A burst read transaction of 80 bytes is shown in Figure 3-5. This example is for an AXI Master Burst core configured for a 32-bit native data width of 32-bits and a maximum allowed burst length of 16 data beats per AXI4 transaction. The command length of 80 bytes requires the master to break the transaction up into two AXI4 transactions, one of 16 data beats and one of four data beats.

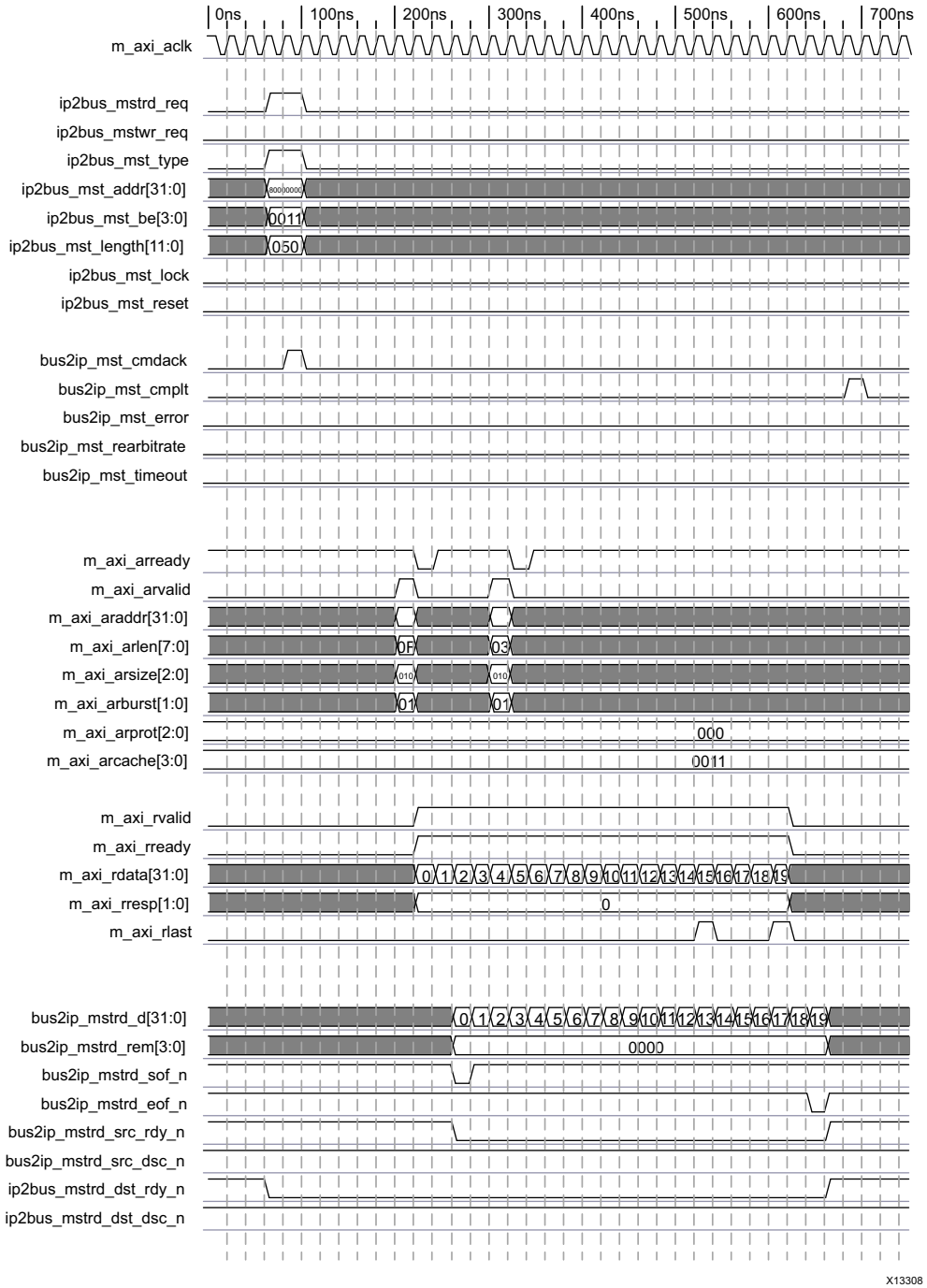


*Figure 3-5:* **Example Burst Read Transaction Timing**

## Burst Read Discontinue

The AXI Master Burst core issues a discontinue on the Read LocalLink if an internal error is encountered during the read transaction. This is normally caused by the User logic setting the `ip2bus_mst_length` qualifier to a value of zero on the IPIC Command interface during a read command assertion. LocalLink requires all transactions to complete with an EOF assertion, even during a discontinue. Figure 3-6. shows an example of the AXI Master Burst core issuing a discontinue on a Read burst transaction as the result of an internal error. The Read LocalLink is terminated early with the EOF assertion by the source.

If the LocalLink is not terminated correctly by the destination, the AXI Master Burst core does not assert the `bus2ip_mst_cmplt` status signal.

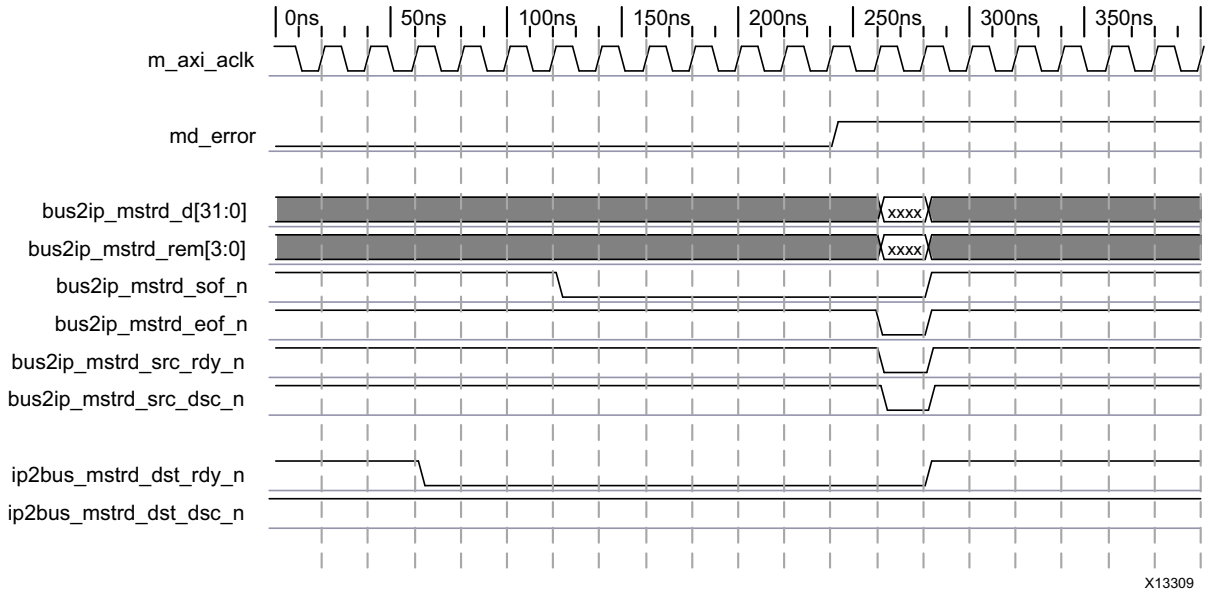


*Figure 3-6:* **Example Burst Read Discontinue Timing**

## Burst Write Transaction

A burst write transaction of 80 bytes is shown in Figure 3-7. This example is for a AXI Master Burst core configured for a 32-bit native data width and a maximum allowed burst length of 16 data beats per AXI4 transaction. The command length of 80 bytes requires the master to break the transaction up into two AXI4 transactions, one 16 data beats and one four data beats.

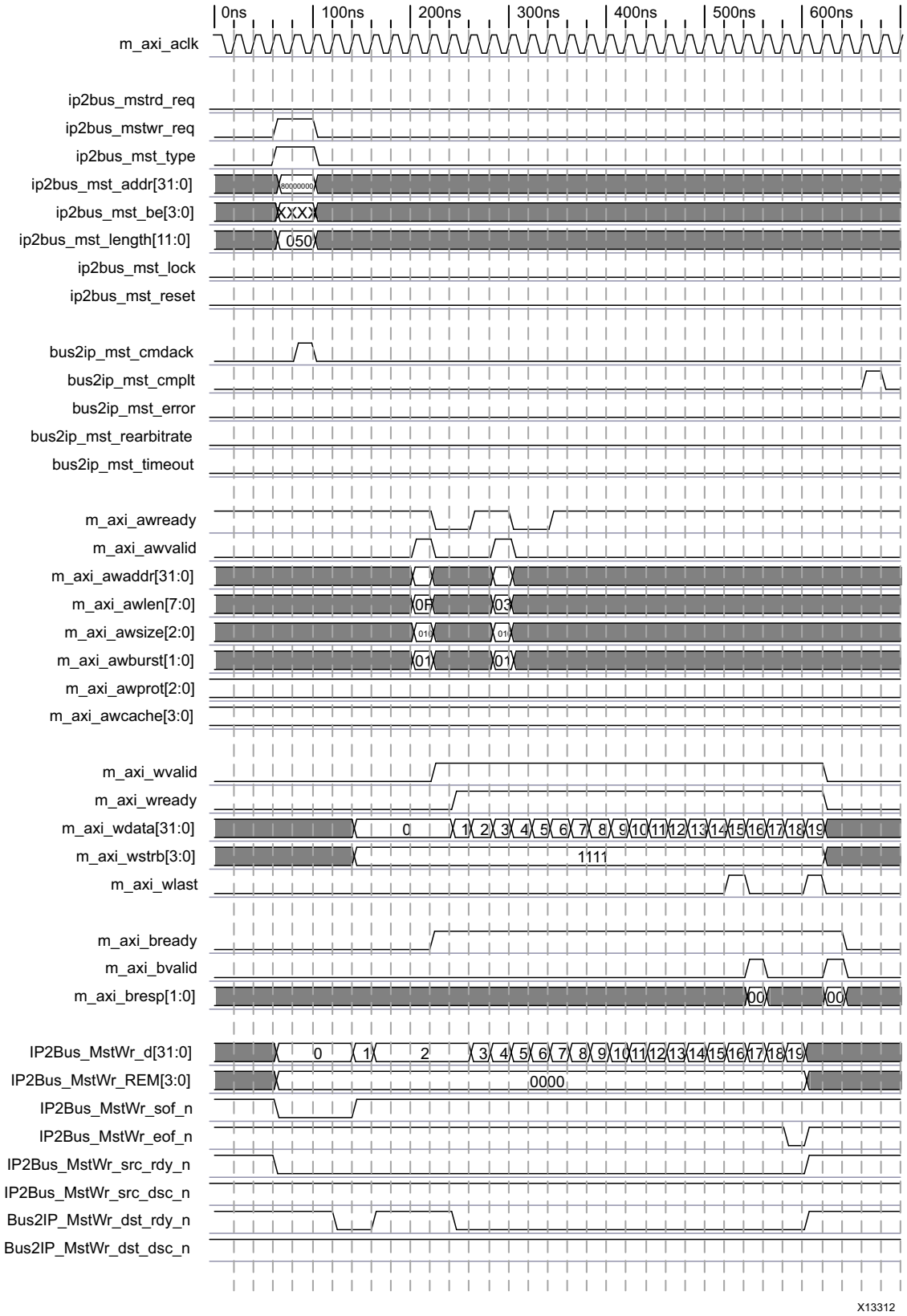


*Figure 3-7:* **Example Burst Write Transaction Timing**

## Burst Write Discontinue

The AXI Master burst issues a discontinue on the Write LocalLink if an internal error is encountered during the write transaction. This is normally caused by the User logic setting the `ip2bus_mst_length` qualifier to a value of zero on the IPIC Command interface during a write command assertion. LocalLink requires all transactions to complete with an EOF assertion, even during a discontinue. Figure 3-8. shows an example of the AXI Master Burst core issuing a discontinue on a Write burst transaction as the result of an internal error. The Write LocalLink is terminated early with the EOF assertion by the source after the `bus2ip_mstwr_dst_dsc_n` assertion is detected.

If the LocalLink is not terminated correctly by the source, the AXI Master Burst core does not assert the `bus2ip_mst_cmplt` status signal.

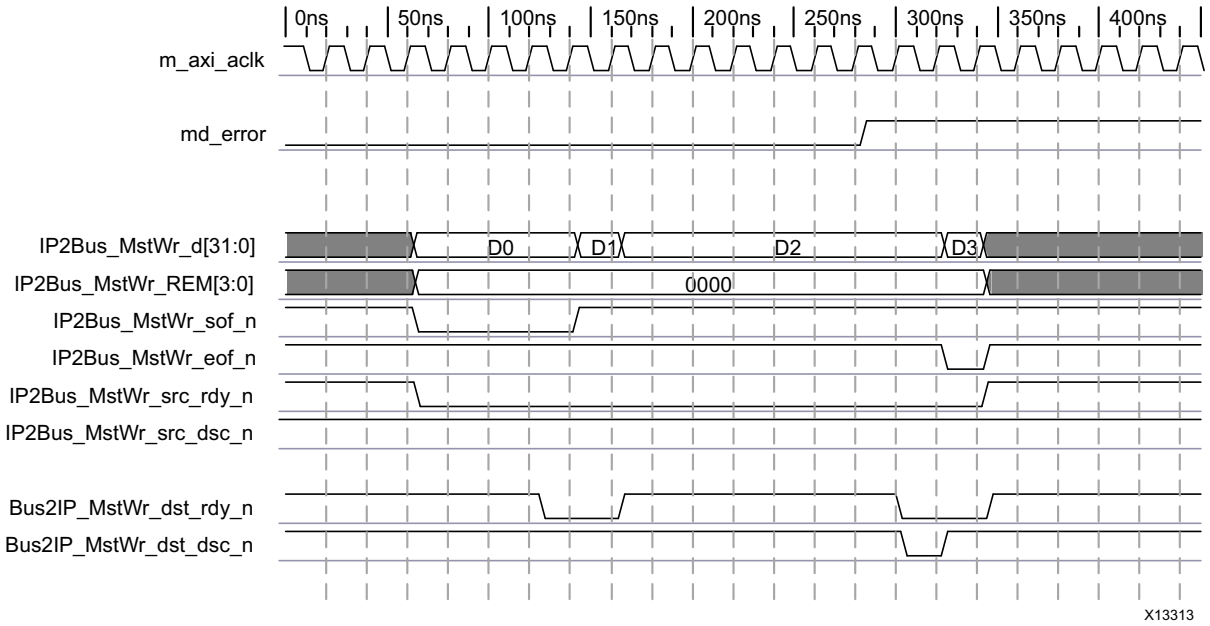


*Figure 3-8:* **Example Burst Write Discontinue Timing**

# Customizing and Generating the Core

The AXI Master Burst core is a helper core, so no specific core graphical user interface is available. The core needs to be instantiated in the user design and package through Vivado® IP Packager. The interface is available for the user design, which can optionally include parameters for configuring the AXI Master Burst core.

# Constraining the Core

This chapter contains information about constraining the AXI Master Burst core in the Vivado® Design Suite environment.

## Required Constraints

No information is currently provided for this core.

## Device, Package, and Speed Grade Selections

No information is currently provided for this core.

## Clock Frequencies

This core works with the AXI4 clock only.

# Simulation

For comprehensive information about Vivado® simulation components, as well as information about using supported third-party tools, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 2].

# Synthesis and Implementation

For details about synthesis and implementation, see "Synthesizing IP" and "Implementing IP" in the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 5].

# Example Design

No example design is provided with this core.

# Test Bench

No test bench is provided with this core.

# Debugging

This appendix includes details about resources available on the Xilinx Support website and debugging tools.

## Finding Help on Xilinx.com

To help in the design and debug process when using the AXI Master Burst, the Xilinx Support web page (www.xilinx.com/support) contains key resources such as product documentation, release notes, answer records, information about known issues, and links for opening a Technical Support WebCase.

### Documentation

This product guide is the main document associated with the AXI Master Burst core. This guide, along with documentation related to all products that aid in the design process, can be found on the Xilinx Support web page (www.xilinx.com/support) or by using the Xilinx Documentation Navigator.

Download the Xilinx Documentation Navigator from the Design Tools tab on the Downloads page (www.xilinx.com/download). For more information about this tool and the features available, open the online help after installation.

### Answer Records

Answer Records include information about commonly encountered problems, helpful information on how to resolve these problems, and any known issues with a Xilinx product. Answer Records are created and maintained daily ensuring that users have access to the most accurate information available.

Answer Records for this core are listed below, and can also be located by using the Search Support box on the main Xilinx support web page. To maximize your search results, use proper keywords such as

- Product name
- Tool message(s)

- Summary of the issue encountered

A filter search is available after results are returned to further target the results.

**Answer Record for the AXI Master Burst Core**

AR 55043

## Contacting Technical Support

Xilinx provides technical support at www.xilinx.com/support for this LogiCORE™ IP 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.

To contact Xilinx Technical Support:

1. Navigate to www.xilinx.com/support.
2. Open a WebCase by selecting the WebCase link located under Support Quick Links.

When opening a WebCase, include:

- Target FPGA including package and speed grade.
- All applicable Xilinx Design Tools and simulator software versions.
- Additional files based on the specific issue might also be required. See the relevant sections in this debug guide for guidelines about which file(s) to include with the WebCase.

# Debug Tools

There are many tools available to address AXI Master Burst design issues. It is important to know which tools are useful for debugging various situations.

## Vivado Lab Tools

Vivado inserts logic analyzer and virtual I/O cores directly into your design. Vivado Lab Tools allows you to set trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be analyzed. This feature represents the functionality in the Vivado IDE that is used for logic debugging and validation of a design running in Xilinx FPGA devices in hardware.

The Vivado logic analyzer is used to interact with the logic debug LogiCORE IP cores, including:

- ILA 2.0 (and later versions)
- VIO 2.0 (and later versions)

### Reference Boards

Various Xilinx development boards support AXI Master Burst. These boards can be used to prototype designs and establish that the core can communicate with the system.

- 7 series FPGA evaluation boards
  - KC705
  - KC724
- Zynq-7000 SoC evaluation boards
  - ZC702
  - ZC706

## Hardware Debug

Hardware issues can range from link bring-up to problems seen after hours of testing. This section provides debug steps for common issues. The Vivado Lab Tools are a valuable resource to use in hardware debug. The signal names mentioned in the following individual sections can be probed using the Vivado Lab Tools for debugging the specific problems.

Many of these common issues can also be applied to debugging design simulations.

### General Checks

Ensure that all the timing constraints for the core were properly incorporated from the example design and that all constraints were met during implementation.

- Does it work in post-place and route timing simulation? If problems are seen in hardware but not in timing simulation, this could indicate a PCB issue. Ensure that all clock sources are active and clean.
- If using MMCMs in the design, ensure that all MMCMs have obtained lock by monitoring the `LOCKED` port.
- If your outputs go to 0, check your licensing.

*Appendix B*

# Additional Resources

## Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see the Xilinx Support website at:

www.xilinx.com/support.

For a glossary of technical terms used in Xilinx documentation, see:

www.xilinx.com/company/terms.htm.

## References

These documents provide supplemental material useful with this product guide:

1. *ARM® AMBA® AXI Protocol v2.0*
2. *Vivado Design Suite User Guide - Logic Simulation* (UG900)
3. *Vivado Design Suite User Guide - Implementation* (UG904)
4. *ISE to Vivado Design Suite Migration Guide* (UG911)
5. *Vivado Design Suite User Guide: Designing with IP* (UG896)

## Revision History

The following table shows the revision history for this document.

| Date | Version | Revision |
|---|---|---|
| 03/20/2013 | 1.0 | Initial Xilinx release as a product guide. Based on DS844. |
| 12/18/2013 | 2.0 | • Updated documented to include UltraScale™ architecture support information.<br>• Added Simulation, Synthesis and Implementation, Example Design, and Test Bench information.<br>• Updated document version number to align with core version number. |

# Notice of Disclaimer