LogiCORE IP AXI DataMover v5.0

Product Guide for Vivado Design Suite

PG022 March 20, 2013

XILINX®

XILINX

Table of Contents

Introduction

The Advanced eXtensible Interface (AXI) DataMover is a soft Xilinx LogiCORE™ Intellectual Property (IP) core that provides the basic AXI4 Read to AXI4-Stream and AXI4-Stream to AXI4 Write data transport and protocol conversion. The function is intended to be a standalone core for custom designs.

Features

- AXI4 Compliant
- Primary AXI4 data width support of 32, 64, 128, 256, 512, and 1024 bits
- Primary AXI4-Stream data width support of 8, 16, 32, 64, 128, 256, 512 and 1024 bits
- Parameterized Memory Map Burst Lengths of 2, 4, 8, 16, 32, 64, 128, and 256 data beats
- Optional Unaligned Address access
- Optional General Purpose Store-And-Forward in both Memory Map to Stream (MM2S) and Stream to Memory Map (S2MM)
- Optional Indeterminate Bytes to Transfer (BTT) mode in S2MM
- Supports synchronous/asynchronous clocking for Command/Status interface

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family(1)	Zynq™-7000, Virtex®-7, Kintex™-7, Artix™-7
Supported User Interfaces	AXI4, AXI4-Stream
Resources	See 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	Mentor Graphics Questa® SIM Vivado Simulator
Synthesis	Vivado Synthesis
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 DataMover is a key interconnect infrastructure IP that enables high throughput transfer of data between the AXI4 memory-mapped and AXI4-Stream domains. The AXI DataMover provides the MM2S and S2MM AXI4-Stream channels that operate independently in a full duplex like method. The AXI DataMover IP core is a key building block with 4 KB address boundary protection, automatic burst partitioning, and provides the ability to queue multiple transfer requests using nearly the full bandwidth capabilities of the AXI4-Stream protocol. Furthermore, the AXI DataMover provides byte-level data realignment allowing memory reads and writes to any byte offset location.

Figure 1-1 and Figure 1-2 show block diagrams of the AXI DataMover core. There are two sub blocks:

- MM2S: This block handles transactions from the AXI memory map to AXI4-Stream domain. It has its dedicated AXI4-Stream compliant command and status queues, reset block and error signals. Based on command inputs, the MM2S block issues a read request on the AXI memory map interface. Read data can be optionally stored inside the MM2S block. Datapath interfaces (AXI4-Read and AXI4-Stream Master) can optionally be made asynchronous to command and status interfaces (AXI4-Stream Command and AXI4-Stream Status).
- S2MM: This block handles transactions from the AXI4-Stream to AXI memory map domain. It has its dedicated AXI4-Stream compliant command and status queues, reset block and error signals. Based on command inputs and input data from the AXI4-Stream interface, the S2MM block issues a write request on the AXI memory map interface. Input stream data can be optionally stored inside a S2MM block. Datapath interfaces (AXI4-Read and AXI4-Stream Master) can optionally be made asynchronous to command and status interfaces (AXI4-Stream Command and AXI4-Stream Status).

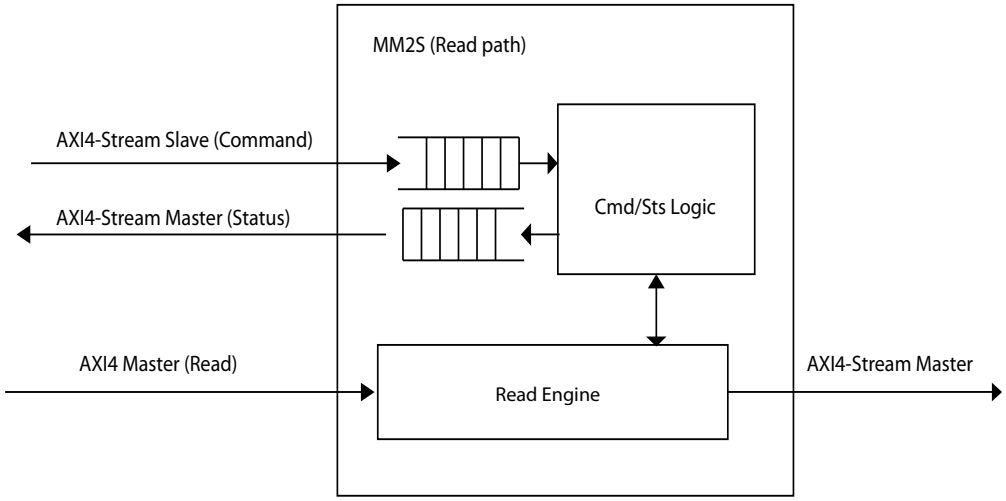

Figure 1-1: **AXI DataMover Read Path**

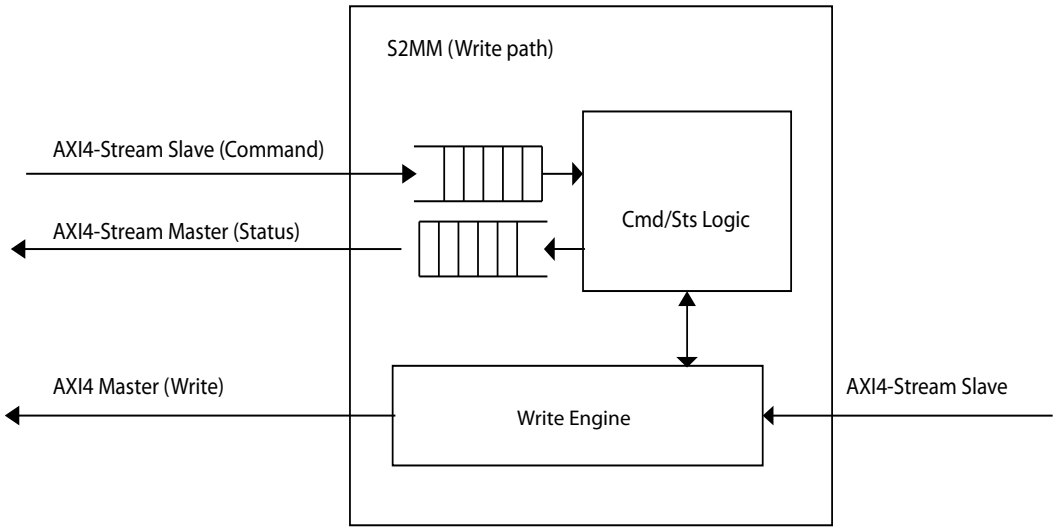

Figure 1-2: **AXI DataMover Write Path**

Feature Summary

AXI4 Compliant

The AXI DataMover core is fully compliant with the AXI4 interface and the AXI4-Stream interface.

AXI4 Data Width

The AXI DataMover core supports the primary AXI4 data bus width of 32, 64, 128, 256, 512, and 1024 bits.

AXI4-Stream Data Width

The AXI DataMover core supports the primary AXI4-Stream data bus width of 8, 16, 32, 64, 128, 256, 512, and 1024 bits. The AXI4-Stream data width must be less than or equal to the AXI4 data width for the respective channel.

Maximum Memory Map Burst Length

The AXI DataMover core supports parameterized maximum size of the burst cycles on the AXI MM2S Memory Map interface. In other words, this setting specifies the granularity of burst partitioning. For example, if the burst length is set to 16, the maximum burst on the memory map interface is 16 data beats. Smaller values reduce throughput but result in less impact on the AXI infrastructure. Larger values increase throughput but result in a greater impact on the AXI infrastructure. Valid supported values are 2, 4, 8, 16, 32, 64, 128, and 256.

Unaligned Transfers

The AXI DataMover core supports optional the Data Realignment Engine (DRE). When DRE is enabled, the DRE allows data realignment to the byte (8 bits) level on the Memory Map datapath. DRE support is provided up to 64 bits TDATA width of AXI4-Stream interface.

If the DRE is enabled, data reads can start from any Buffer Address byte offset, and the read data is aligned such that the first byte read is the first valid byte out on the AXI4-Stream. Similarly, when the DRE is enabled, the writes can happen at any byte offset address. What is considered aligned or unaligned is based on the Memory Map data width. For example, if Memory Map Data Width = 32, data is aligned if it is located at address offsets of 0x0, 0x4, 0x8, 0xC, and so forth. Data is unaligned if it is located at address offsets of 0x1, 0x2, 0x3 and so forth.

Asynchronous Clocks

The AXI DataMover core supports asynchronous clock domain for Command/Status Stream interface and Memory Map interface.

Store and Forward

The AXI DataMover core supports the optional General Purpose Store-And-Forward feature. Store and Forward buffer is sized based on Memory map data width and Burst size support. It is sized to allow up to six outstanding requests on MM2S channel.

Indeterminate BTT Mode

The AXI DataMover core supports the optional Indeterminate BTT mode for the S2MM channel. This is needed when the number of bytes to be received on the input S2MM Stream Channel is unknown.

Applications

The AXI DataMover provides high-speed data movement between system memory and an AXI4-Stream-based target. This core is intended to be a standalone core for a custom design.

Unsupported Features

The following AXI4 features are not supported by the DataMover design.

- User signals
- Locking transfers
- Caching transfers
- Non-incrementing and circular Burst transfers

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

Performance

The AXI Datamover is characterized as per the benchmarking methodology described in Appendix A, IP Characterization and fMAX Margin System Methodology, *Vivado Design Suite User Guide: Designing with IP* (UG896). Table 2-1 shows the results of the characterization runs.

Table 2-1: **Maximum Frequencies**

Family	Speed Grade	FMAX (MHz) All Clocks
Virtex-7	-1	200
Kintex-7		180
Artix-7		150
Virtex-7	-2	220
Kintex-7		200
Artix-7		160
Virtex-7	-3	250
Kintex-7		220
Artix-7		170

Latency

Table 2-2 describes the latency for the AXI DataMover core. Latency is measured in simulation and indicates AXI DataMover core latency cycles only and does not include system dependent latency or throttling.

Table 2-2: **AXI DataMover Latency**

Description	Clocks
MM2S Channel	
Initial m_axi_mm2s_rvalid to m_axis_mm2s_tvalid (Store and Forward disabled)	1
Initial m_axi_mm2s_rvalid to m_axis_mm2s_tvalid (Store and Forward enabled)	3
AXI4-Stream packet to packet latency (No Unaligned access) m_axis_mm2s_tlast to m_axis_mm2s_tvalid	2
AXI4-Stream packet to packet latency (Unaligned access) m_axis_mm2s_tlast to m_axis_mm2s_tvalid	3
s_axis_mm2s_cmd_tvalid to m_axi_mm2s_arvalid	8
S2MM Channel	
Initial s_axis_s2mm_tvalid m_axi_s2mm_awalid (Store and Forward disabled)	2
Initial s_axis_s2mm_tvalid m_axi_s2mm_awalid (Store and Forward disabled)	20
AXI4-Stream packet to packet latency (No unaligned access) s_axis_s2mm_tlast to s_axis_s2mm_tready	2
AXI4-Stream packet to packet latency (Unaligned access enabled) s_axis_s2mm_tlast to s_axis_s2mm_tready	3

Throughput

Table 2-3 describes the latency for the AXI DataMover core. The tables provides performance information for a typical configuration. Throughput test consisted of eight parent commands loaded into the AXI DataMover core with each command having BTT value as 1 MB and each channel operating simultaneously (full duplex). The core was configured for synchronous operation meaning `m_axis_mm2s_cmdsts_aclk` = `m_axis_s2mm_cmdsts_awclk` = `m_axi_mm2s_aclk` = `m_axi_s2mm_aclk`.

The Core configuration used to generate the throughput data is as follows:

- Memory Map Data Width = 32
- Streaming Data Width = 32
- No unaligned access
- Store and Forward enabled

Table 2-3: **AXI DataMover Throughput**

AXI DataMover Channel	Primary Clock Frequency	Packet Size	Maximum Total Data Throughput (MB/sec)	Percent of Theoretical
MM2S	100	1 MB	391.27	97.75%
S2MM	100	1 MB	391.27	97.75%

Resource Utilization

The AXI DataMover resource utilization for various parameter combinations measured with 7 series and Zynq™-7000 devices. See Table 2-4.

Table 2-4: **7-Series and Zynq-7000 Device Resource Estimates**

MM2S								S2MM									Resource			
Type	MMap Width	Streaming Width	Burst Length	BTT Width	Async Mode	Store and Forward	Unaligned Transfer	Type	MMap Width	Streaming Width	Burst Length	BTT Width	Async Mode	Store and Forward	Unaligned Transfer	Indeterminate BTT	Slice	LUT	REG	Block RAM
Basic	32	32	16	16	NA	NA	NA	Basic	32	32	16	16	NA	NA	NA	NA	199	413	715	0
Full	32	32	16	16	FALSE	TRUE	FALSE	Full	32	32	16	16	FALSE	TRUE	FALSE	FALSE	416	1028	1042	2
Full	64	64	8	23	FALSE	TRUE	FALSE	Full	64	64	8	23	FALSE	TRUE	FALSE	FALSE	543	1315	1390	3
Full	64	32	8	16	FALSE	TRUE	TRUE	Full	64	32	8	16	FALSE	TRUE	TRUE	FALSE	529	1387	1461	3
Full	32	32	16	16	TRUE	TRUE	FALSE	Full	32	32	16	16	TRUE	FALSE	FALSE	TRUE	532	1174	1595	6

Port Descriptions

The AXI DataMover I/O signals are described in Table 2-5.

Table 2-5: **I/O Signal Description**

Signal Name	Interface	Signal Type	Init Status	Description
Memory Map to Stream Clock and Reset				
m_axi_mm2s_aclk	MM2S	Input	–	Master Clock for MM2S path
m_axi_mm2s_aresetn	MM2S	Input	–	Master Reset for the MM2S logic. Active-Low assertion sensitivity. Must be asserted for three clock periods of m_axi_mm2s_aclk.
Memory Map to Stream Soft Shutdown Control				
mm2s_halt	MM2S	Input	–	Active-High input signal requesting that the MM2S function perform a soft shutdown and stop. See MM2S Soft Shutdown.
mm2s_halt_cmplt	MM2S	Output	0	Active-High output signal indicating that the MM2S function has completed a soft shutdown and is stopped. See MM2S Soft Shutdown.
Memory Map to Stream Error Detect				
mm2s_err	MM2S	Output	0	MM2S Error Output. This active-High output discrete signal is asserted whenever an Error condition is encountered within the MM2S such as an invalid BTT value of 0. This bit is a “sticky” error indication; after being set it requires an assertion of the m_axi_mm2s_aresetn signal to clear it.

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

Signal Name	Interface	Signal Type	Init Status	Description
Memory Map to Stream Debug Support				
mm2s_dbg_sel(3:0)	MM2S	Input	–	Reserved for internal Xilinx use.
mm2s_dbg_data(31:0)	MM2S	Output	BEEF0000 (if Omit MM2s) BEEF1111 (if Full MM2s) BEEF2222 (if Basic MM2s)	Reserved for internal Xilinx use.
Memory Map to Stream Address Posting Control and Status				
mm2s_allow_addr_req	MM2S	Input	–	Used to control the MM2S in posting an address on the AXI4 Read address channel. A "1" allows posting and a "0" inhibits posting. See Address Posting Control and Status Interface.
mm2s_addr_req_ posted	MM2S	Output	0	This output signal is asserted to "1" for one m_axi_mm2s_aclk period for each new address posted to the AXI4 Read Address Channel. See Address Posting Control and Status Interface.
mm2s_rd_xfer_cmplt	MM2S	Output	0	This output signal is asserted to 1 for one m_axi_s2mm_aclk period for each completed AXI4 read transfer (qualified RLAST data beat) clearing the internal read data controller block.
AXI4 Read Interface Signals				
m_axi*	M_AXI_MM2S	–	–	See Appendix A of the *AXI Reference Guide* (UG761) for the description of AXI4 Signals.
AXI4-Stream Master Interface Signals				
m_axis*	M_AXIS*	–	–	See Appendix A of the *AXI Reference Guide* (UG761) for the description of AXI4 Signals.

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

Signal Name	Interface	Signal Type	Init Status	Description
Memory Map to Stream Command/Status Channel Asynchronous Clock and Reset				
m_axis_mm2s_ cmdsts_aclk	MM2S Command & Status	Input	–	MM2S Command Interface Clock. This clock is only used if asynchronous clocks are enabled in the Vivado IDE. The frequency of this clock is expected to be equal or less than the m_axi_mm2s_aclk.
m_axis_mm2s_ cmdsts_aresetn	MM2S Command & Status	Input	–	MM2S Command and Status Interface Reset (Active-Low). This reset input is only used if asynchronous clocks are enabled in the Vivado IDE. Must be asserted for three clock periods of m_axis_mm2s_cmdsts_aclk.
AXI4 Slave Stream Interface Signals				
s_axis*	S_AXIS*	Input/ Output	Input/ Output	See Appendix A of the *AXI Reference Guide* (UG761) for the description of AXI4 Signals.
Stream to Memory Map Clock and Reset				
m_axi_s2mm_aclk	S2MM	Input	–	Master Clock for S2MM path
m_axi_s2mm_aresetn	S2MM	Input	–	Master Reset for the S2MM logic (Active-Low sensitivity). Must be asserted for three clock periods of m_axi_s2mm_aclk.
Stream to Memory Map Soft Shutdown Control				
s2mm_halt	S2MM	Input	–	Active-High input signal requesting that the S2MM function perform a soft shutdown and stop. See S2MM Soft Shutdown.
s2mm_halt_cmplt	S2MM	Output	0	Active-High output signal indicating that the S2MM function has completed a soft shutdown and is stopped. See S2MM Soft Shutdown.
Stream to Memory Map Error Detect				
s2mm_err	S2MM	Output	0	S2MM Error Output. This active-High output discrete signal is asserted whenever an Error condition is encountered within the S2MM such as an invalid BTT of 0 or a Stream overrun or underrun when S2MM Indeterminate BTT is not enabled. This bit is a "sticky" error indication; after being set it requires an assertion of the m_axi_s2mm_aresetn signal to clear it.

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

Signal Name	Interface	Signal Type	Init Status	Description
Stream to Memory Map Debug Support				
s2mm_dbg_sel(3:0)	S2MM	Input	–	Reserved for internal Xilinx use.
s2mm_dbg_data(31:0)	S2MM	Output	CAFE0000 (if Omit S2MM) CAFE1111 (if Full S2MM) CAFE2222 (if Basic S2MM)	Reserved for internal Xilinx use.
Stream to Memory Map Address Posting Control and Status				
s2mm_allow_addr_req	S2MM	Input	–	Used to control the S2MM in posting an address on the AXI4 Write Address Channel. A "1" allows posting and a "0" inhibits posting. See Address Posting Control and Status Interface.
s2mm_addr_req_posted	S2MM	Output	0	This output signal is asserted to "1" for one m_axi_s2mm_aclk period for each new address posted to the AXI4 Write Address Channel.
s2mm_wr_xfer_cmplt	S2MM	Output	0	This output signal is asserted to "1" for one m_axi_s2mm_aclk period for each completed AXI4 write transfer (qualified WLAST data beat) clearing the internal write data controller block.
s2mm_ld_nxt_len	S2MM	Output	0	This output signal is asserted to "1" for one m_axi_s2mm_aclk period for each AXI4 Write Transfer request to be posted to the AXI4 Write Address channel. This reflects internal queue loading so its assertion is prior to it appearing on the Write Address Channel. This signal qualifies the value on the s2mm_wr_len output port for use by external logic.
s2mm_wr_len	S2MM	Output	0	This bus reflects the value that is placed on the m_axi_s2mm_awlen output (AXI4 Write Address Channel) when it is pulled from the internal queue. The value is only valid when the signal s2mm_ld_nxt_len is asserted.
AXI4 Write Interface Signals				
m_axi_s2mm*	M_AXI_S2MM*	–	–	See Appendix A of the *AXI Reference Guide* (UG761) for the description of AXI4 Signals.

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

Signal Name	Interface	Signal Type	Init Status	Description
Stream to Memory Map Command/Status Channel Asynchronous Clock and Reset				
m_axis_s2mm_cmdsts_awclk	S2MM Command & Status	Input	–	S2MM Command Interface Clock. asynchronous clocks are enabled in the Vivado IDE. The frequency of this clock is expected to be equal or less than the m_axi_s2mm_aclk.
m_axis_s2mm_cmdsts_aresetn	S2MM Command & Status	Input	–	S2MM Command Interface Reset (Active-Low). This reset input is only used if asynchronous clocks are enabled in the Vivado IDE. Must be asserted for three clock periods of m_axis_s2mm_cmdsts_awclk.

Design Information

Command Interface

The DataMover operations are controlled by an AXI Slave Stream interface that receives transfer commands from the user logic. The MM2S and the S2MM each have a dedicated command interface. A command is loaded with a single data beat on the input Command Stream interface. The width of the command word is normally 72 bits if 32-bit AXI Addressing is being used in the system. However, the command word width is extended (by parameterization) if the system address space grows beyond 32 bits. For example, a 64-bit address system requires the command word to be 104 bits wide to accommodate the wider starting address field.

The format of the command word is shown in Figure 2-1 and detailed in Table 2-6. It is the same for either the MM2S or S2MM DataMover elements. The command format allows the specification of a single data transfer from 1-byte to 8,388,607 bytes (7FFFFF hex bytes). A command loaded into the command interface is often referred to as the parent command of a transfer. The DataMover automatically breaks up large transfers into intermediate bursts (child transfers) that comply with the AXI4 Memory Mapped protocol requirements.

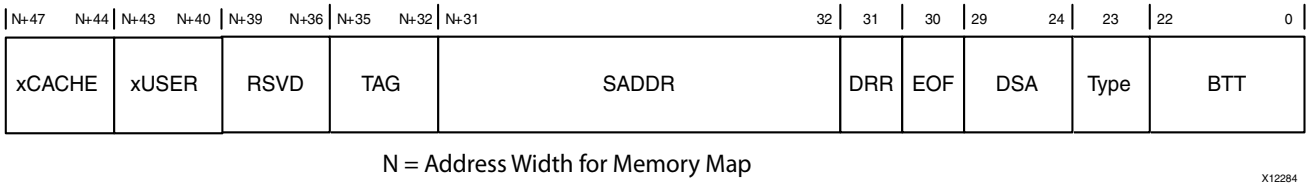

Figure 2-1: **Command Word Layout**

Table 2-6: **Command Word Description**

Bits	Field Name	Description
(N + 47) - (N + 44)[2]	xCACHE	The value written in this field appears on m_axis_mm2s_arcache for MM2S block and m_axis_s2mm_awcache for S2MM block.
(N + 43) - (N + 40)[2]	xUSER	The value written in this field appears on m_axis_mm2s_aruser for MM2S block and m_axis_s2mm_awuser for S2MM block.
(N+39) - (N+36)[1]	RSVD	**Reserved**
(N+35) - (N+32)[1]	TAG	**Command TAG** This field is an arbitrary value assigned by you to the Command. The TAG flows through the DataMover execution pipe and gets inserted into the corresponding status word for the Command.
(N+31) -32[1]	SADDR	**Start Address** This field indicates the starting address to use for the Memory Mapped side of the transfer requested by the command. If DRE is enabled, the lower order address bits of this field indicate the starting alignment to load on the Memory Mapped side of the DRE.
31	DRR	**DRE ReAlignment Request** This bit is only used if the optional DRE is included by parameterization. The bit indicates that the DRE alignment needs to be re-established prior to the execution of the associated command. The DRE Stream side alignment is derived from the DSA field of the command. The Memory Mapped side alignment is derived from the least significant bits of the SADDR field.
30	EOF	**End of Frame** This bit indicates that the command is an End of Frame command. This generally affects the MM2S element (Read Master) because it causes the Stream output logic to assert the TLAST output on the last data beat of the last transfer needed to complete the command. *If DRE is included*, this also causes the DRE to Flush out any intermediate data at the conclusion of the last transfer of the command and submit it to the Stream output (in the case of the MM2S Read Master) or to the AXI Write Data Channel (in the case of the S2MM Write Master).

Table 2-6: **Command Word Description *(Cont'd)***

Bits	Field Name	Description
29-24	DSA	**DRE Stream Alignment** This field is only used by the MM2S and if the optional MM2S DRE is included by parameterization. The field is only used when the DRR bit of the associated command is also set to 1. This 6-bit field indicates the reference alignment of the MM2S Stream Data Channel for the optional DRE. The value is byte-lane relative. A value of 0 indicates byte lane 0 (least significant byte) is the reference byte lane; a value of 1 indicates byte lane 1, and so on. Valid values are dependent upon the parameterized data width of the Stream data Channel. For example, a 32-bit wide data channel has only 4-byte lane positions and thus the DSA field can only have values of 0 to 3. ***Note:*** DRE alignment on the associated Memory Mapped side is derived from the least significant bits of the SADDR value of the command.
23	Type	This field determines the type of AXI4 access. Setting this to 1 enables INCR. A value of 0 enables FIXED address AXI4 transaction.
22 to 0	BTT	**Bytes to Transfer** This 23-bit field indicates the total number of bytes to transfer for the command. A transfer of 1 up to 8,388,607 bytes. A value of 0 is not allowed and causes an internal error from the DataMover. The actual number of BTT bits used by the DataMover is controlled by the parameters 'Width of BTT field' for MM2S and S2MM respectively.

Notes:

1. N is equal to 32, which is the width of Memory Mapped Address bus.
2. These fields are valid only when "Enable xUSERxCACHE" is checked in the Vivado Integrated Design Environment (IDE).

Command FIFO

The Command interface of a DataMover element is designed to allow command queuing. The commands are "queued" in a FIFO.

The Command FIFO is by default a synchronous FIFO clocked by the same clock that is clocking the Memory Mapped Data and Address channels of the associated DataMover element. However, you can specify an asynchronous command interface FIFO. This allows the command interface to be clocked at a different (generally much slower) frequency than the Memory Mapped interface.

Command Stream Interface Timing

Loading a command into the Command FIFO is mechanized by a single AXI4-Stream data beat. An example of loading five commands into the MM2S Command FIFO is shown in Figure 2-2. In this scenario, the Command FIFO is synchronous to the Memory Mapped interface. `TLAST` and `TSTRB` signals are ignored.

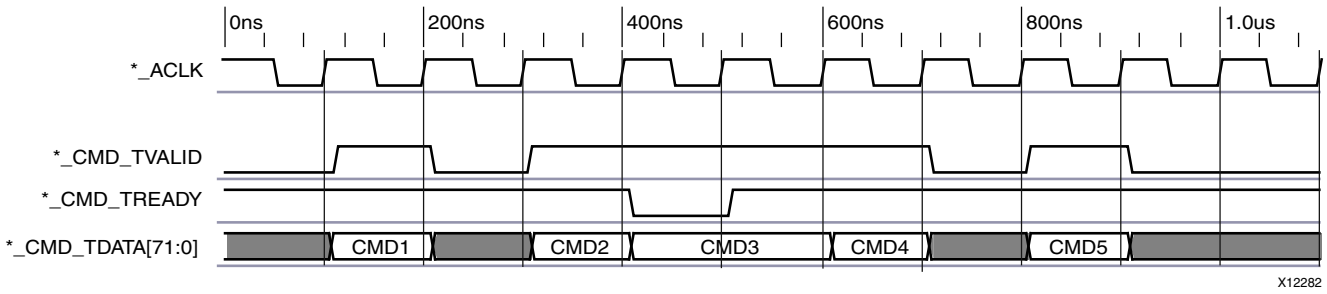

Figure 2-2: **Loading Commands via the Command Interface**

Status Interface

The status of DataMover transfer operations are provided by an AXI Master Stream interface that relays transfer status to the user logic. The MM2S and the S2MM each have a dedicated Status Interface. A status word is read with a single data beat on the Status Stream interface. The width of the status word is fixed at 8 bits except when S2MM is enabled in Indeterminate mode (S2MM Status Format in Indeterminate BTT Mode (IBTT).)

The format of the status word is shown in Figure 2-3 and detailed in Table 2-7. It is the same for either the MM2S or S2MM DataMover elements.

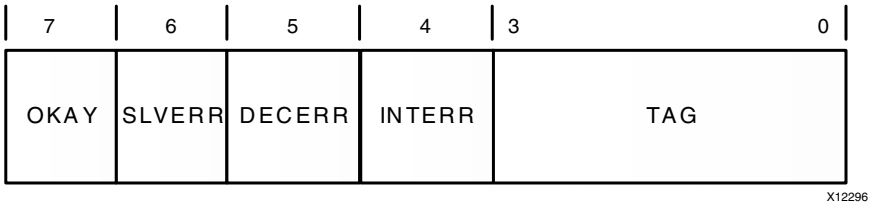

Figure 2-3: **Normal Status Word Layout**

Table 2-7: **Status Word Details**

Bits	Field Name	Description
7	OKAY	**Transfer OKAY** This bit indicates that the associated transfer command has been completed with the OKAY response on all intermediate transfers. 0 = Command had a non-OKAY response during all associated transfers 1 = Command had a OKAY response during all associated transfers
6	SLVERR	**Slave Error** Indicates the DataMover encountered a Slave reported error condition for the associated command. This is received by the Response inputs from the AXI4 Memory Mapped interface. 0 = No Error 1 = Slave Asserted Error Condition

Table 2-7: **Status Word Details *(Cont'd)***

Bits	Field Name	Description
5	DECERR	**Decode Error** Indicates the DataMover encountered an address decode error condition for the associated command. This is received by the Response inputs from the AXI4 Memory Mapped interface and indicates an address decode timeout occurred on an address generated by the DataMover Element while executing the corresponding Command. 0 = No Error 1 = Address Decode Error Condition
4	INTERR	**Internal Error** Indicates the DataMover encountered an internal error condition for the associated command. A BTT (Bytes to Transfer) value of 0 (zero) in the Command Word can cause this assertion. The S2MM function can also assert this if the input stream TLAST assert occurs prematurely (relative to the commanded BTT for the transfer) and the Indeterminate BTT mode is not enabled. 0 = No Error 1 = Internal Error Condition
3 to 0	TAG	**TAG** This 4-bit field echoes the value of the TAG field of the associated input Command whose completion generated the Status.

S2MM Status Format in Indeterminate BTT Mode (IBTT)

The DataMover S2MM function can be parameterized to enable support for Stream data transfer of an indeterminate number of bytes. This is defined as where the S2MM is commanded (by the BTT command field) to transfer a fixed number of bytes, but it is unknown how many bytes are actually going to be received from the incoming Stream interface (at the assertion of `s_axis_s2mm_tlast`). Supporting this operation mode requires additional hardware in the S2MM function, and additional fields in the status word indicating the actual count of the bytes received from the Stream interface for the commanded transfer, and whether the TLAST was received during the transfer.

The format of the S2MM status word with Indeterminate BTT mode enabled is shown in Figure 2-4 and detailed in Table 2-8. This status format does not apply to the MM2S DataMover status interface. See Indeterminate BTT Mode for more information.

Figure 2-4: **S2MM Status Format in IBTT mode**

Table 2-8: **Special S2MM Status Word Details (Indeterminate BTT Mode Enabled)**

Bits	Field Name	Description
31	EOP	**End of Packet** This bit indicates that the S2MM Stream input received a TLAST assertion during the execution of the DataMover command associated with the status word. This is not an error condition. It is needed by certain Users (that is, Scatter Gather Engines) to identify the actual End of Packet for the input Stream versus the theoretical maximum that could occur.
30 to 8	BRCVD	**Bytes Received** This field indicates the actual number of bytes received on the Stream interface at the point where s_axis_s2mm_tlast was asserted by the Stream Master.
7	OKAY	**Transfer OKAY** This bit indicates that the associated transfer command has been completed with the OKAY response on all intermediate transfers. 0 = Command had a non-OKAY response during all associated transfers 1 = Command had a OKAY response during all associated transfers
6	SLVERR	**Slave Error** Indicates the DataMover encountered a Slave reported error condition for the associated command. This is received by the Response inputs from the AXI4 Memory Mapped interface. 0 = No Error 1 = Slave Asserted Error Condition
5	DECERR	**Decode Error** Indicates the DataMover encountered an address decode error condition for the associated command. This is received by the Response inputs from the AXI4 Memory Mapped interface and indicates an address decode timeout occurred on an address generated by the DataMover Element while executing the corresponding Command. 0 = No Error 1 = Address Decode Error Condition
4	INTERR	**Internal Error** Indicates the DataMover encountered an internal error condition for the associated command. A BTT (Bytes to Transfer) value of 0 (zero) in the Command Word can cause this assertion. The S2MM function can also assert this if the input stream TLAST assert occurs prematurely (relative to the commanded BTT for the transfer) and the Indeterminate BTT mode is not enabled. Additional conditions are To Be Determined. 0 = No Error 1 = Internal Error Condition
3 to 0	TAG	**TAG** This 4-bit field echoes the value of the TAG field of the associated input Command whose completion generated the Status.

Status FIFO

The Status Interface of a DataMover is designed to allow for status queuing corresponding to the available command queuing on the Command Interface. The status values are "queued" in a FIFO.

The Status FIFO is by default a synchronous FIFO clocked by the same clock that is clocking the Memory Mapped interface of the associated DataMover element. However, you can specify an asynchronous Command/Status Interface FIFO. This allows the Command and Status Interfaces to be clocked at a different (generally much slower) frequency than the associated Memory Mapped interface.

Status Read Interface Timing

Reading a status word from the Status FIFO is mechanized by a single AXI4-Stream data beat. An example of reading five status entries from the MM2S Status FIFO is shown in Figure 2-5. In this scenario, the Status FIFO is synchronous to the Memory Mapped Address and Data Channel clock. The example illustrates that a status word is considered read from the Status FIFO only when both the TVALID and TREADY handshake signals are both asserted at the rising edge of the synchronizing clock.

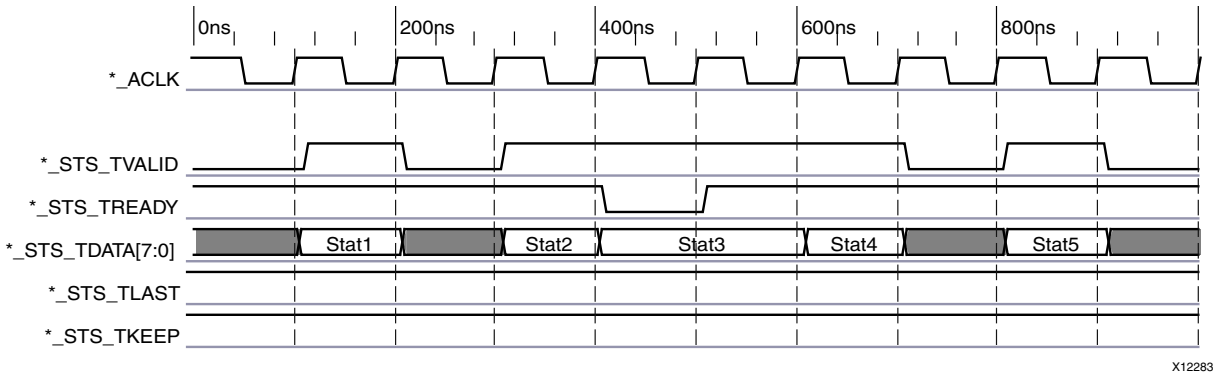

Figure 2-5: **Reading Status over the Status Interface**

MM2S Store and Forward

The MM2S can include an optional Store and Forward block when the parameter Enable Store Forward is enabled. Enabling this parameter ensures that child transfers are not posted to the AXI4 Read Address Channel if there is not enough space left in the Store and Forward FIFO for the data.

S2MM Store and Forward

The S2MM can include an optional Store and Forward block when the parameter Enable Store Forward is enabled. Enabling this parameter ensures that transfers are not posted to the AXI4 Write Address Channel until all of the data needed for the requested transfer is present in the Store and Forward FIFO.

Indeterminate BTT Mode

The AXI DataMover S2MM function has a special operating mode to support the case when the amount of data being received in the Stream Channel is unknown (or indeterminate).

An additional feature of the Indeterminate BTT mode is the absorption of overflow data from the input Stream channel. Overflow is defined as the stream data that is received that exceeds the BTT value for the corresponding parent transfer command and the EOF bit is also set in that command. The data absorption occurs from the point of the BTT value being reached to the next TLAST data beat.

The corresponding status output by the S2MM block for the associated transfer command does not have the EOP bit set and the BRCVD field in the status word only reflects the commanded BTT value, not the actual number of bytes received for the input overflow packet. Only the data up to the BTT value is written to the Memory Mapped space by the S2MM AXI4 Write Data Channel.

Address Posting Control and Status Interface

The AXI Data Mover provides additional Address Posting Control and its Status signals that allows you to exercise control over initiating transactions on Memory Map side even corresponding commands already were issued on the command interface. These signals can be controlled by user logic to minimize the need for AXI DataMover to throttle the AXI Memory Map Interface.

These ports are:

- `mm2s_allow_addr_req` (input to DataMover)
- `mm2s_addr_req_posted` (output from DataMover)
- `mm2s_rd_xfer_cmplt` (output from DataMover)
- `s2mm_allow_addr_req` (input to DataMover))
- `s2mm_addr_req_posted` (output from DataMover)
- `s2mm_wr_xfer_cmplt` (output from DataMover)
- `s2mm_ld_nxt_len` (output from DataMover)
- `s2mm_wr_len` (output from DataMover)

Note: These signals are irrelevant for most of the use cases and are classified for advance use. By default the access to these signals is disabled. You can gain access to these signals by enabling them in the customization Vivado™ IDE

Example Design

The connection of these ports to an external user logic is shown in Figure 2-6. This is representative of the loopback connection on the AXI4-Stream side with external storage facility. The user logic should have the ability to control the AXI DataMover Address posting to the AXI4 bus through the `mm2s_allow_addr_req` and `s2mm_allow_addr_req` signals. When asserted high, the associated DataMover Address Controller is allowed to post transfer address to the AXI4 bus and thus commit to a transfer. The `mm2s_allow_addr_req` controls the MM2S Address Controller and the `s2mm_allow_addr_req` controls the S2MM Address Controller. When asserted low, the associated Address Controller is inhibited from posting transfer address to the AXI4 bus.

The AXI DataMover also provides status back to the user logic indicating when an address set has been committed to the AXI4 bus through the `mm2s_addr_req_posted` and `s2mm_addr_req_posted` signals. In addition, the MM2S and S2MM also provide a status bit indicating when a scheduled Read or Write Data Channel transfer has completed through the `mm2s_rd_xfer_cmplt` and `s2mm_wr_xfer_cmplt` signals.

The S2MM function provides two additional outputs (`s2mm_wr_len` and `s2mm_ld_nxt_len`) that are used to provide some lookahead to the user monitoring logic by indicating the needed data beats for each of the upcoming Write Transfers that are being queued in the S2MM Write Data Controller. By monitoring the input stream from the MM2S, the user monitoring logic can count the incoming data and notify the write side monitor logic when the exact amount of data has been received to satisfy a queued write transfer.

This control and status mechanism allows the AXI DataMover to pipeline Read requests to the AXI4 without over-committing the Data FIFO capacity (filling it up and throttling the AXI4 Read data Channel). It also can keep the DataMover from issuing Write transfers until the write data is actually present in the Data FIFO. See Timing Diagrams in Figure 2-8 and Figure 2-9 for a better understanding of usage of these signals.

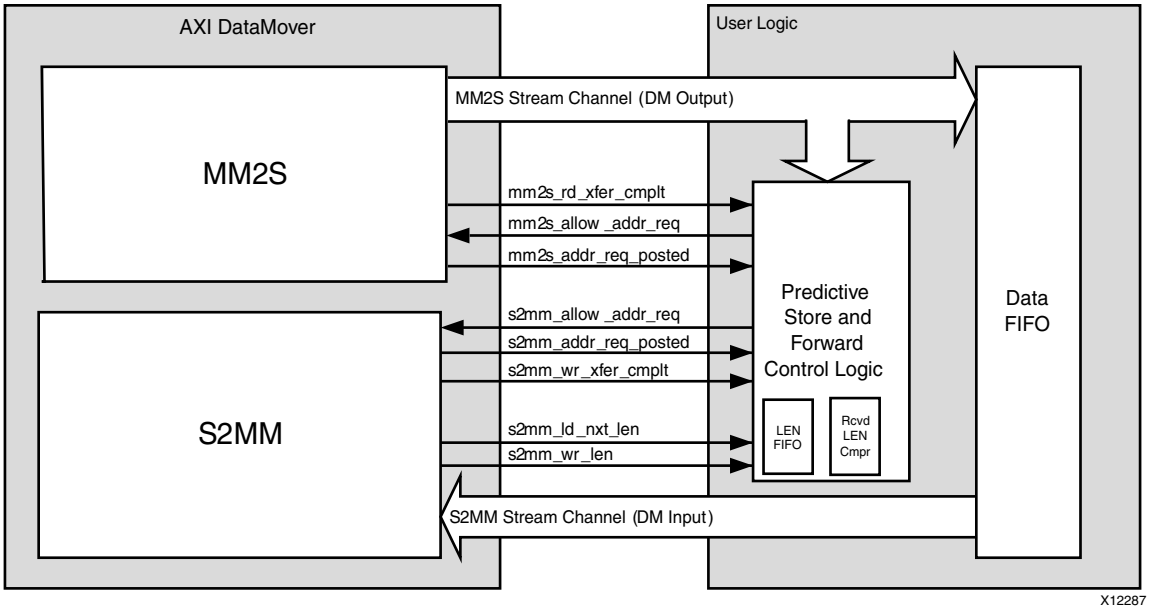

Figure 2-6: **AXI DataMover Address Posting Control and Status Interface Use Case**

Request Spawning

One important aspect of the DataMover operation is the ability to spawn multiple child AXI requests when executing a single command from the corresponding Command FIFO. This occurs when the requested Bytes to Transfer (BTT) specified by the Command exceeds a parameterized burst data beat limit (default is 16 but can also be set to 32. 64, 128, or 256).

MM2S Soft Shutdown

The DataMover MM2S soft shutdown is initiated by the active-High assertion of the input signal `mm2s_halt`. During a soft shutdown, DataMover gracefully completes the existing transactions on the AXI MM side. You will find data on the streaming side sometime while its exiting gracefully. The `m_axis_mm2s_tdata` data output is then driven with invalid data values. The MM2S completes all committed Memory Map requests presented on the MM2S Memory Map Address Channel. Input data from the Memory Map Data Channel received during the cleanup operations are discarded.

When the soft halt operations are completed, the MM2S asserts the `mm2s_halt_cmplt` output. This output remains asserted until the MM2S is reset through the hard reset input `m_axi_mm2s_aresetn` (or `m_axis_mm2s_cmdsts_aresetn` if asynchronous command interface is in use).

S2MM Soft Shutdown

The S2MM soft shutdown is initiated by the active-High assertion of the input signal `s2mm_halt`. During a soft shutdown, the S2MM function asserts the S2MM Stream `s_axis_s2mm_tready` output signal and ignores the remaining S2MM Stream inputs. The S2MM completes all committed Memory Map requests presented on the S2MM Memory Map Address Channel. Output data to the Memory Map Data Channel transmitted during the cleanup operations are invalid data values.

When the soft halt operations are completed, the S2MM asserts the `s2mm_halt_cmplt` output. This output remains asserted until the S2MM is reset by the hard reset input `m_axi_s2mm_aresetn` (or `m_axis_s2mm_cmdsts_aresetn` if the asynchronous command interface is in use).

DataMover Basic

Some applications of the DataMover do not need the high performance features it provides. In these applications, resource utilization is more important than performance. The DataMover provides the ability to select a reduced function option through the Vivado IDE.

The following feature simplifications characterize the Basic version:

- 32-bit and 64-bit Memory Mapped Data Width and 8, 16, 32, and 64-bit Stream width (parameterized). Starting transfer address must be aligned to address boundaries that are multiples of the Stream Data width (in bytes).
- Maximum AXI4 Burst Length support of 16, 32, and 64 data beats (parameterized).
- No DRE support.
- One-Deep Command and Status Queuing (Parent command). The Command and Status FIFOs are replaced with a FIFO register for each.

- Commanded transfer lengths (Bytes to Transfer) are limited to the Max AXI4 Burst Length multiplied by the Stream data width (in bytes).

  Example: Maximum burst length = 32, Stream Data Width = 4 bytes (32 bits), the maximum commanded transfer length (BTT) is 128 bytes.

- No breakup of transfers into smaller bursts.
- 4K byte boundaries are not monitored.
- Automatic transfer splitting at an AXI 4K address boundary is not supported.
- No Store and Forward support.

DataMover Basic Command Interface

The format of the Basic command word is shown in Figure 2-7 and detailed in Table 2-9.

Note: * = S2MM or MM2S

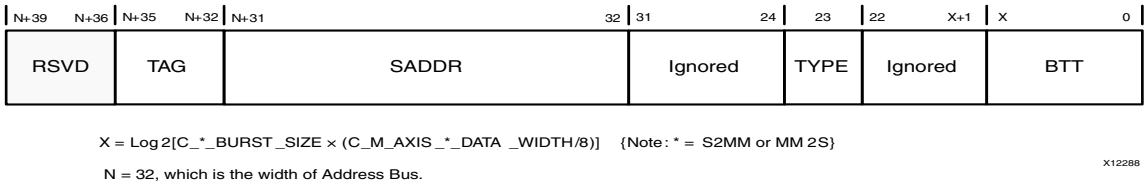

Figure 2-7: **DataMover Basic Command Word Layout**

Table 2-9: **DataMover Basic Command Word Details**

Bits	Field Name	Description
(N+39) - (N+36)[(1)]	RSVD	**Reserved** This field is reserved to pad the command width to an even multiple of 8 bits. (required for AXI4-Stream interfaces)
(N+35) - (N+32)[(1)]	TAG	**Command TAG** You assign this field an arbitrary value to the Command. The TAG flows through the DataMover execution pipe and gets inserted into the corresponding status word for the Command.
(N+31) -32[(1)]	SADDR	**Start Address** This field indicates the starting address to use for the Memory Mapped side of the transfer requested by the command.
31-24	Ignored	This field is ignored by the DataMover Basic. Can be any value but zeroes are recommended.
23	TYPE	**TYPE** Specifies the type of AXI4 access. A '1' means INCR access, while '0' means a FIXED address access.
22-(X+1)	Ignored	This field is ignored by the DataMover Basic. Can be any value but zeroes are recommended.

Table 2-9: **DataMover Basic Command Word Details *(Cont'd)***

Bits	Field Name	Description
X-0	BTT	**Bytes to Transfer** This field indicates the total number of bytes to transfer for the command. The maximum allowed value is set by the following formula: Max Burst Length x (Streaming Data width)/8

Notes:

1. N is equal to 32, which is the width of Address Memory Mapped Address bus.

DataMover Basic Status Interface

The format of the status word is the same as the full version and is shown in Figure 2-3, and detailed in Table 2-7.

Example DataMover Read(MM2S) Timing

Figure 2-8 illustrates example timing on read (MM2S) path in synchronous mode.

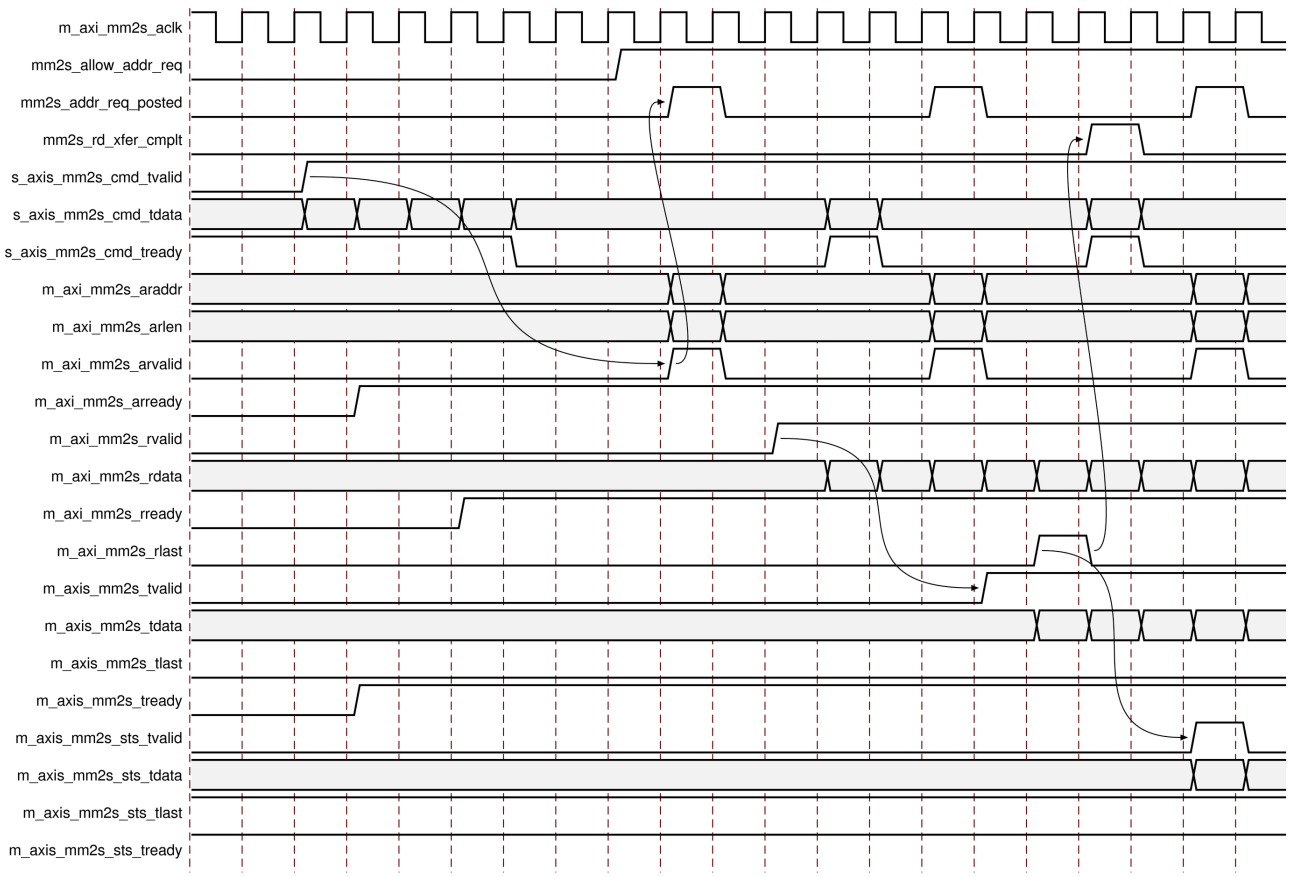

Figure 2-8: **Example Timing on Read (MM2S) Path in Synchronous Mode**

Dataflow:

1. After receiving commands on the AXI4-Stream command interface (`s_axis_mm2s_cmd_tvalid`) and if `mm2s_allow_addr_req` is high, AXI DataMover initiates read cycle on the AXI MMap interface by asserting `m_axi_mm2s_arvalid` and other address bus signals.
2. It also asserts `mm2s_addr_req_posted` indicating address is posted on the MMap interface.
3. Read data is stored in internal FIFO if enabled.
4. AXI DataMover starts sending out data on the streaming interface by asserting `m_axis_mm2s_tvalid` and other associated signals.
5. AXI DataMover asserts `mm2s_rd_xfer_cmplt` indicating data is completely read on the MMap interface.
6. AXI4-Stream Status interface signals `m_axis_mm2s_sts_tvalid` and other associated signals are asserted indicating the status for a particular command that was posted on command interface

IMPORTANT: *A single parent command can generate multiple child commands on the AXI MMap Interface. Status signals are asserted when all child commands are processed.*

Note: In the absence of any S2MM command, AXI Datamover will pull the `s_axis_s2mm_tready` signal to low. This will throttle the input data stream. To have a minimum amount of throttling, ensure that a valid command is issued to the S2MM interface much before the actual data arrives.

Example DataMover Write(S2MM) Timing

Figure 2-9 illustrates example timing on write (S2MM) path in synchronous mode.

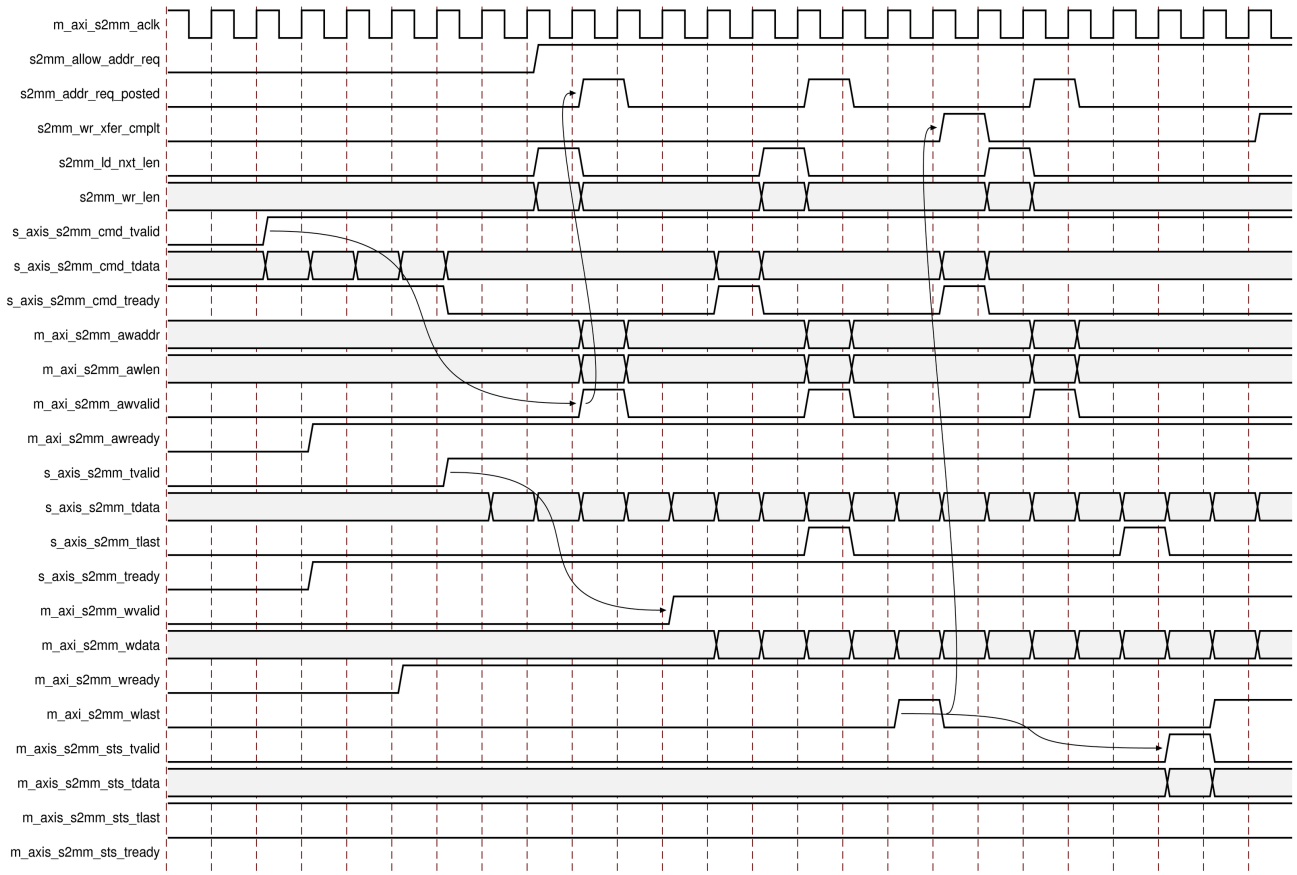

Figure 2-9: **Example Timing on Write (S2MM) Path in Synchronous Mode**

Dataflow:

1. After receiving commands on the AXI4-Stream command interface (`s_axis_s2mm_cmd_tvalid`) and if `s2mm_allow_addr_req` is high, AXI DataMover initiates write cycles on the AXI MMap interface by asserting `m_axi_s2mm_awvalid` and other address bus signals.
2. AXI DataMover also asserts `mm2s_addr_req_posted` indicating address is posted on MMap interface.
3. AXI DataMover accepts data on the streaming interface by asserting `s_axis_s2mm_tready`.
4. Incoming data is stored in FIFO if enabled.
5. AXI DataMover starts sending out data on MMap interface by asserting `m_axi_s2mm_wvalid` and other associated signals.

6. AXI DataMover asserts `s2mm_wr_xfer_cmplt` indicating data is completely written on the MMap interface.
7. AXI4-Stream Status interface signals `m_axis_s2mm_sts_tvalid` and other associated signals are asserted indicating the status for a particular command that was posted on command interface.
8. AXI DataMover also asserts additional signals `s2mm_ld_nxt_len` along with `s2mm_wr_len` indicating the burst length of the write transfer to be posted on the AXI MMap interface.

IMPORTANT: *A single parent command can generate multiple child commands on the AXI MMap Interface. Status signals are asserted when all child commands are processed.*

Designing with the Core

Figure 3-1 and Figure 3-2 show typical use cases of AXI DataMover. Figure 3-1 shows a multichannel application of DataMover in the MM2S path.

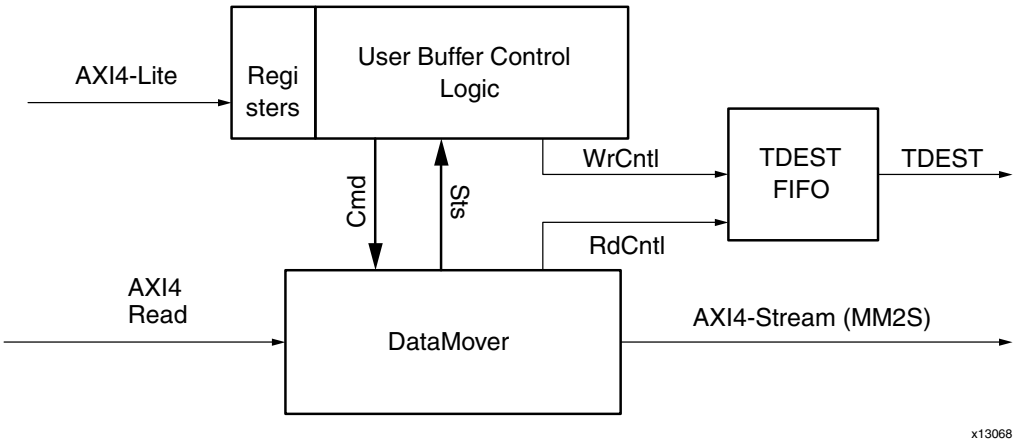

Figure 3-1: **Typical Application of MM2S DataMover**

Figure 3-2 shows multichannel application of DataMover in S2MM path. Incoming TDEST information can be used to pick the corresponding destination address on the AXI MM side and same TDEST value can be stored in the register space.

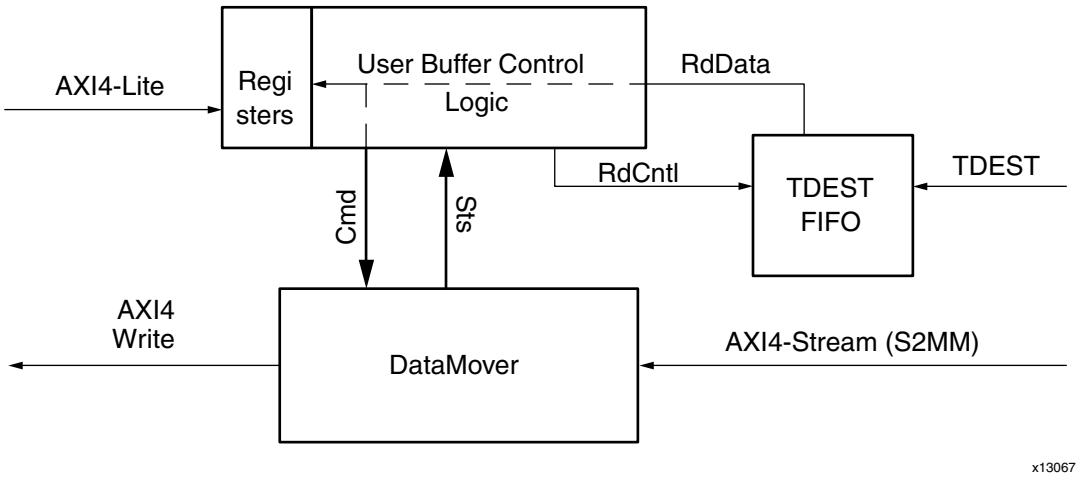

Figure 3-2: **Typical Application of S2MM DataMover**

Clocking

The AXI DataMover has two clock inputs for each of the MM2S and S2MM blocks for a total of four clock inputs. The `m_axi_mm2s_aclk` is the main synchronizing clock for the MM2S block. This clock synchronizes both the associated Memory Mapped interface and Stream interface. The second clock for the MM2S element is `m_axis_mm2s_cmdsts_awclk`. This clock is used only when MM2S is configured in asynchronous mode. When used, it synchronizes the User sides of the Command and Status interfaces.

The S2MM block has identical clocking schemes as the MM2S block but with two different clocks, the `m_axi_axi_s2mm_aclk` and `m_axis_s2mm_cmdsts_awclk`.

Resets

The AXI DataMover has two reset inputs for each of the MM2S and S2MM blocks for a total of four reset inputs.

AXI DataMover requires that input reset assertion must be a minimum of three clock periods of the synchronizing clock. Table 3-1 shows clock and reset signals and its associated interface.

Table 3-1: **Clock, Reset and its Associated Interface**

		Asynchronous Mode	
Blocks	**Interface**	**Disabled**	**Enabled**
MM2S	Memory map and Streaming Interface	m_axi_mm2s_aclk and m_axi_mm2s_aresetn	m_axi_mm2s_aclk and m_axi_mm2s_aresetn
	Command and Status Interface	m_axi_mm2s_aclk and m_axi_mm2s_aresetn	m_axis_mm2s_cmdsts_aclk and m_axis_mm2s_cmdsts_aresetn
		Asynchronous Mode	
Blocks	**Interface**	**Disabled**	**Enabled**
S2MM	Memory map and Streaming Interface	m_axi_s2mm_aclk and m_axi_s2mm_aresetn	m_axi_s2mm_aclk and m_axi_s2mm_aresetn
	Command and Status Interface	m_axi_s2mm_aclk and m_axi_s2mm_aresetn	m_axis_s2mm_cmdsts_aclk and m_axis_s2mm_cmdsts_aresetn

Customizing and Generating the Core

This chapter includes information about using Xilinx tools to customize and generate the core in the Vivado™ Design Suite.

The AXI DataMover can be found in **\AXI_Infrastructure** and also in **Embedded_Processing\AXI_Infrastructure** in the IP catalog.

To access the AXI DataMover, perform the following:

1. Open a project by selecting **File** then **Open Project** or create a new project by selecting **File** then **New Project** in the Vivado design tools.
2. Open the IP catalog and navigate to any of the taxonomies.
3. Double-click on **AXI DataMover** to bring up the **AXI DataMover** Vivado IDE.

The AXI DataMover Vivado IDE contains one screen with two tabs (Figure 4-1 and Figure 4-2) that provide information about the core, allow configuration of the core, and provides the ability to generate the core.

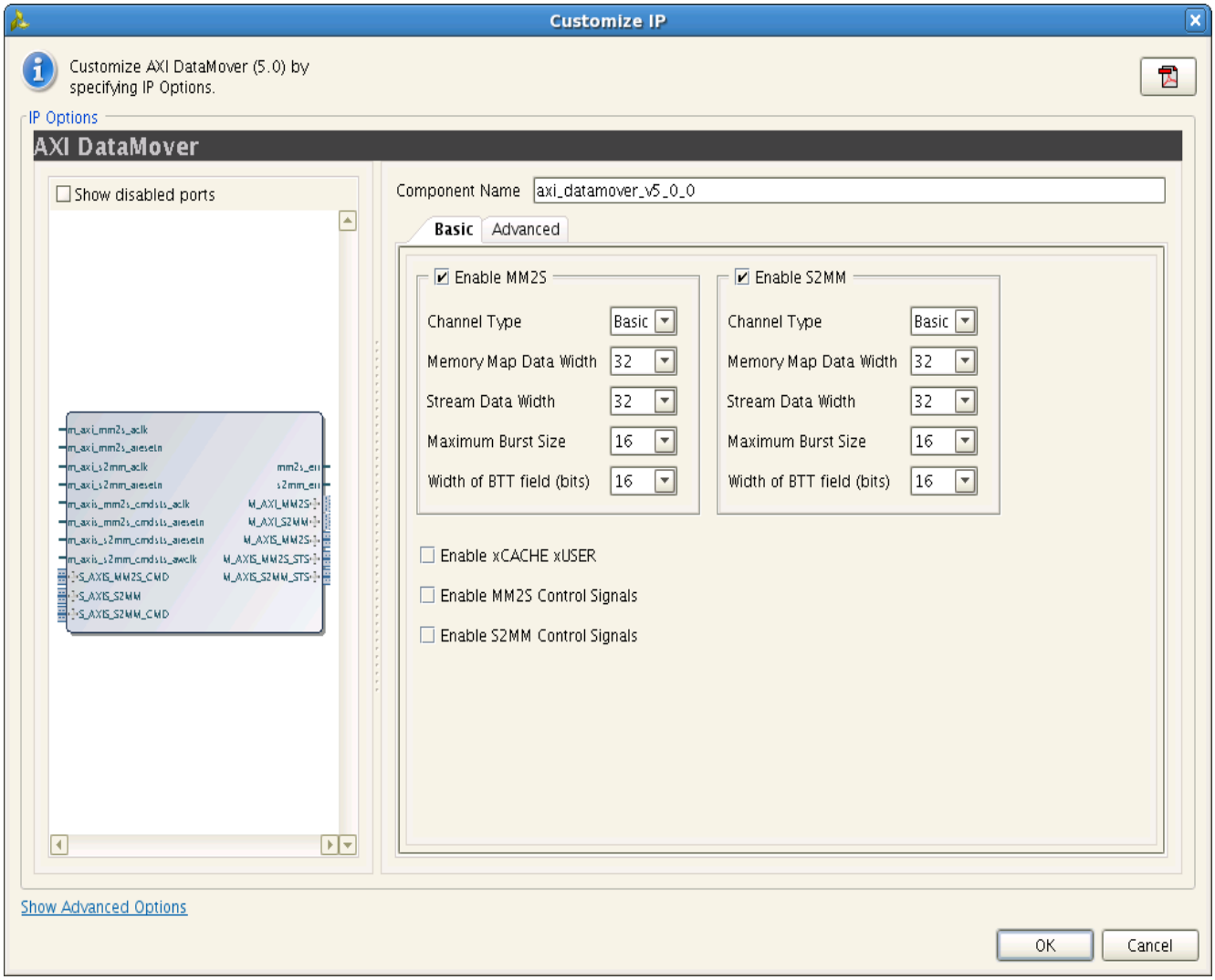

Figure 4-1: **Basic Tab**

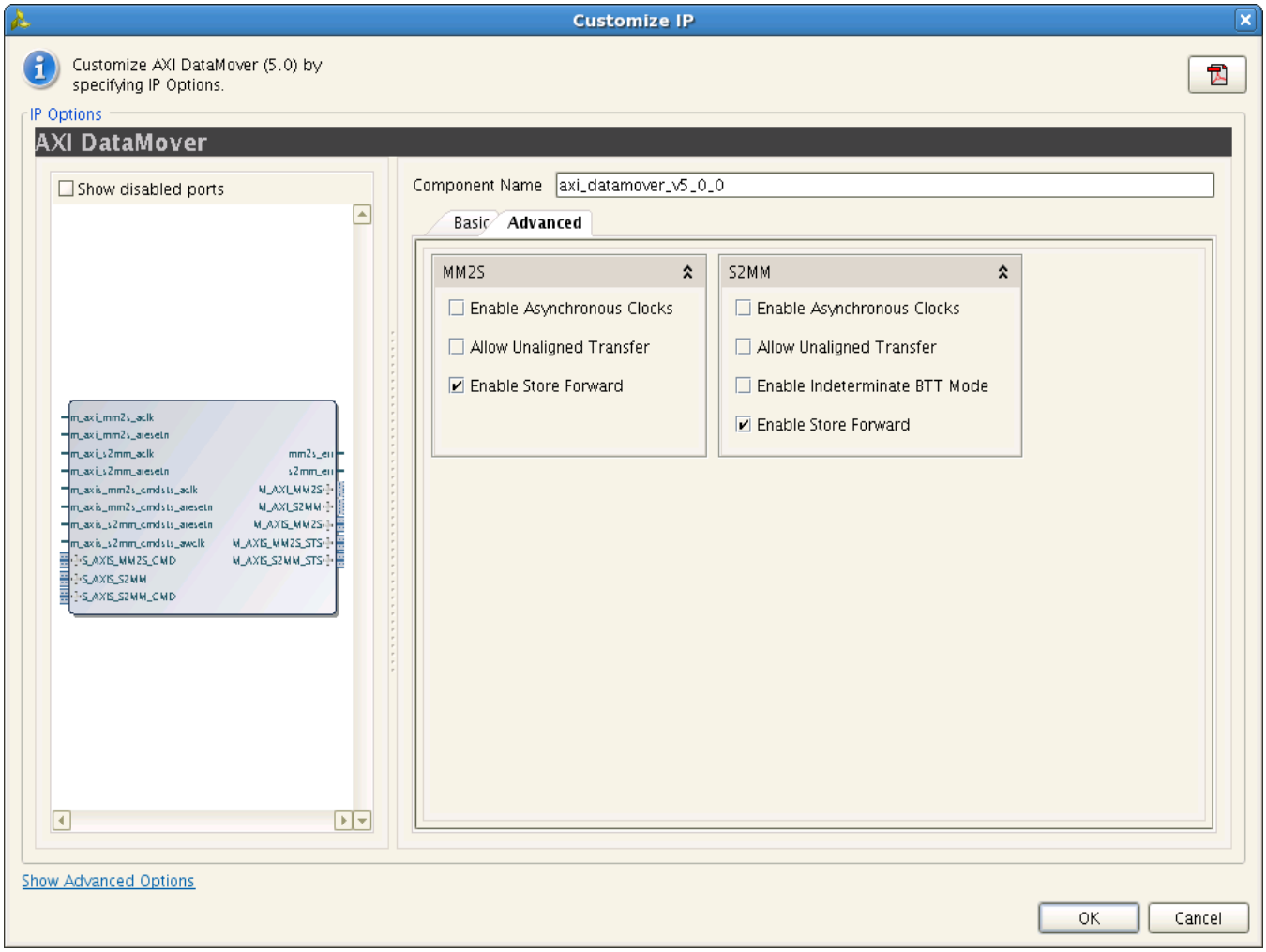

Figure 4-2: **IP Options - Advanced Tab**

Component Name – The base name of the output files generated for the core. Names must begin with a letter and can be composed of any of the following characters: a to z, 0 to 9, and "_".

Basic Options

The following describes the fundamental options that affect the MM2S and S2MM channels of the AXI DataMover core.

- **MM2S Channel Options** – This box allows you to configure the MM2S channel options.
  - **Channel Type** – You can choose a Full or Basic. Selecting Full allows the MM2S channel to be configured for all possible combination and advance features. The Basic mode restricts some of features and allows the MM2S to be used only for 32 or 64-bit wide data.

- **Memory Mapped Data Width** – Specifies the data width in bits of the AXI Memory Mapped Read bus. Valid values are 32, 64, 128, 256, 512, and 1024. Depending on the Channel Type, these options vary.
- **Stream Data Width** – Specifies the data width in bits of the MM2S Stream bus. Valid values are 8, 16, 32, 64, 128, 256, 512, and 1024. This value cannot be more than the Memory Mapped Data Width.
- **Maximum Burst Size** – This option specifies the maximum size of the burst cycles on the AXI MM2S Memory Map Read interface. In other words, this setting specifies the granularity of burst partitioning. For example, if the burst length is set to 16, the maximum burst on the memory map interface is 16 data beats. Smaller values reduce throughput but result in less impact on the AXI infrastructure. Larger values increase throughput but result in a greater impact on the AXI infrastructure. Valid values are 2, 4, 8, 16, 32, 64, 128, and 256.
- **Bytes To Transfer (BTT) Bit Used** – Specifies the valid number of bits in the number of BTT field of the MM2S command. Valid options are 8 to 23.

- **S2MM Channel Options** – This box allows you to configure the S2MM channel options.
  - **Channel Type** – You can choose a Full, Basic, or Omit. Selecting Full allows the S2MM channel to be configured for all possible combination and advance features. The Basic mode restricts some of features and allows the S2MM to be used only for 32 or 64-bit wide data. The Omit mode completely disables the channel.
  - **Memory Mapped Data Width** – Specifies the data width in bits of the AXI Memory Mapped Write bus. Valid values are 32, 64, 128, 256, 512, and 1024. The choices vary depending on the Channel Type chosen.
  - **Stream Data Width** – Specifies the data width in bits of the S2MM Stream bus. Valid values are 8, 16, 32, 64, 128, 256, 512, and 1024. This value cannot be more than the Memory Mapped Data Width.
  - **Maximum Burst Size** – This option specifies the maximum size of the burst cycles on the AXI S2MM Memory Map Read interface. In other words, this setting specifies the granularity of burst partitioning. For example, if the burst length is set to 16, the maximum burst on the memory map interface is 16 data beats. Smaller values reduce throughput but result in less impact on the AXI infrastructure. Larger values increase throughput but result in a greater impact on the AXI infrastructure. Valid values are 2, 4, 8, 16, 32, 64, 128, and 256.
  - **Bytes To Transfer (BTT) Bit Used** – Specifies the valid number of bits in the number of BTT field of the S2MM command. Valid options are 8 to 23.
  - **Enable xCache and xUser** - Select this if you wish to change the *cache and *user signals of the AXI4 interface.

- **Enable MM2S Control Signals, Enable S2MM Control Signals** - Enabling this exposes all the control and status signals of the MM2S, S2MM interface. When enabled, you need to connect the signals carefully. By default, these are not exposed and tied to default states.

Advanced Options

The following describes the advanced options of the MM2S and S2MM channels of the AXI DataMover core.

- **MM2S Channel Options** – This box allows you to configure the advance options of the MM2S channel options.

The following options are only available when the channel is configured in "Full" mode.

- **Enable Asynchronous Clocks** – This setting allows you to operate the MM2S Command and Status Stream interface asynchronously with MM2S Memory Map interface.
- **Allow Unaligned Transfers** – This setting enables or disables the MM2S Data Realignment Engine (DRE). When checked, the DRE is enabled and allows data realignment to the byte (8 bits) level on the MM2S Memory Map datapath. For the MM2S channel, data is read from the memory. If the DRE is enabled, data reads can start from any Buffer Address offset, and the read data is aligned such that the first byte read is the first valid byte out on the AXI4-Stream. What is considered aligned or unaligned is based on the Memory Map data width**.**
- **Enable Store and Forward** – This setting provides the inclusion/omission of the MM2S Store and Forward function.

The following is a list of S2MM Channel parameters.

- **S2MM Channel Options** – This box allows you to configure the advance options of the S2MM channel options.

The following options are only available when the channel is configured in "Full" mode.

- **Enable Asynchronous Clocks** – This setting allows you to operate the S2MM Command and Status Stream interface asynchronously with S2MM Memory Map interface.
- **Allow Unaligned Transfers** – This option enables or disables the S2MM Data Realignment Engine (DRE). When checked, the DRE is enabled and allows data realignment to the byte (8 bits) level on the S2MM Memory Map datapath. For the S2MM channel, data is written to the memory. If the DRE is enabled, data writes can start from any Buffer Address offset, and the read data is aligned such that the first byte read is the first valid byte out on the AXI4-Stream. What is considered aligned or unaligned is based on the Memory Map data width**.**

- **Enable Indeterminate BTT Mode** – This setting provides the Indeterminate BTT mode. This is needed when the number of bytes to be received on the input S2MM Stream Channel is unknown at the time the transfer command is posted to the DataMover S2MM command input.
- **Enable Store and Forward** – This setting provides the inclusion/omission of the S2MM Store and Forward function.

Chapter 5

Constraining the Core

The necessary XDC constraints are delivered when the IP is generated.

Appendix A

IP Migration

For information on migrating to the Vivado™ Design Suite, see *Vivado Design Suite Migration Methodology Guide* (UG911).

Debugging

This appendix includes details about resources available on the Xilinx Support website and debugging tools. In addition, this appendix provides a step-by-step debugging process and a flow diagram to guide you through debugging the AXI DataMover core.

The following topics are included in this appendix:

Finding Help on Xilinx.com

To help in the design and debug process when using the AXI DataMover core, 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 DataMover 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.

Known Issues

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 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.

Master Answer Record for the AXI DataMover Core

AR 54434

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.

Vivado Lab Tools

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

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)

Hardware Debug

- What value should be driven on `mm2s_allow_addr_req` and `s2mm_allow_addr_req` if you do not want to use these signals?

  Answer: `mm2s_allow_addr_req` and `s2mm_allow_addr_req` should be tied to '1' if you do not want to control it. There are example timing diagrams (Fig 3-9 and Fig 3-10) to provide more clarity on usage of these signals. By default, these pins are not exposed and are tied to '1'.

- When `mm2s_halt` is asserted, `m_axis_mm2s_tvalid` is asserted sometimes.

  Answer: When AXI DataMover goes through soft shutdown, it flushes internal FIFOs, thus you will find some residual data appearing on streaming side.

Appendix C

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

To search for Xilinx documentation, go to www.xilinx.com/support

Unless otherwise noted, IP references are for the product documentation page.

These documents provide supplemental material useful with this product guide:

1. *LogiCORE IP AXI Interconnect Product Guide* (PG059)
2. *Vivado Design Suite Migration Methodology Guide* (UG911)
3. Vivado design tools user documentation
4. *AMBA® AXI4-Stream Protocol Specification*
5. *ARM® AXI4 Memory Mapped Specification*
6. *ARM AXI4-Stream Interface Specification*
7. *Vivado Design Suite User Guide: Designing with IP* (UG896)
8. *AXI Reference Guide* (UG761)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
10/19/2011	1.0	Initial Xilinx release.
07/11/2012	1.1	Template update.
07/25/2012	2.0	Updated for Vivado 2012.2, Zynq features, and ISE v14.2 Added Vivado content in Customizing and Generating the Core
10/16/2012	2.0.1	• Updated for Vivado 2012.3 and ISE v14.3. • Added MM2S and S2MM block Information • Added two figures showing typical use cases for DataMover • Removed AXI Read Master, AXI Write Master sections, AXI DataMover Operation, and Parameter -- I/O Signal Dependencies sections • Added two new sections to Chapter 3: Example DataMover Read(MM2S) Timing Example DataMover Write(S2MM) Timing
12/18/2012	2.1	• Updated for Vivado 2012.4 and ISE v14.4 design tools. • Updated Debugging appendix. Updated core version. • Replaced Figure 1-1 with two new figures. • Updated max frequency numbers and devices. • Removed many rows from resource utilization tables. • Removed Allowable Parameter Combinations section. • Updated screen captures. • Updated output hierarchies.
03/20/2013	5.0	• Updated for Vivado design tools and core version 5.0. • Removed all ISE, Virtex-6, and Spartan-6 material. • Removed Design Parameters section in Chapter 3. • Added Type field and Maximum Burst Size option. • Updated screen captures in Chapter 4. • Updated many of the I/O signals.

Notice of Disclaimer