AXI DMA v7.1

LogiCORE IP Product Guide

Vivado Design Suite

PG021 (v7.1) June 24, 2025

Table of Contents

Chapter 1

Introduction

The AMD LogiCORE™ IP AXI Direct Memory Access (AXI DMA) core is a soft AMD IP core for use with the AMD Vivado™ Design Suite. The AXI DMA provides high-bandwidth direct memory access between memory and AXI4-Stream target peripherals. Its optional scatter/gather capabilities also offload data movement tasks from the central processing unit (CPU).

Features

- AXI4 compliant
- Optional Scatter/Gather Direct Memory Access (DMA) support
- AXI4 data width support of 32, 64, 128, 256, 512, and 1,024 bits
- AXI4-Stream data width support of 8, 16, 32, 64, 128, 256, 512, and 1,024 bits
- Optional Keyhole support
- Optional Data Re-Alignment support for streaming data widths up to 512 bits
- Optional AXI Control and Status Streams
- Optional Micro DMA Support
- Support for up to 64-bit addressing

IP Facts

AMD LogiCORE™ IP Facts Table	
Core Specifics	
Supported Device Family[1]	AMD Versal™ Adaptive SoCs, AMD UltraScale+™ AMD UltraScale™, AMD Zynq™ 7000 SoC, 7 series FPGAs
Supported User Interfaces	AXI4, AXI4-Lite, AXI4-Stream
Resources	Performance and Resource Use web page
Provided with Core	
Design Files	VHDL
Example Design	VHDL
Test Bench	VHDL
Constraints File	Delivered with IP Generation
Supported S/W Driver[2]	Standalone and Linux
Tested Design Flows[3]	
Design Entry	AMD Vivado™ Design Suite
Simulation	For supported simulators, see the *Vivado Design Suite User Guide: Release Notes, Installation, and Licensing* (UG973).
Synthesis	Vivado Synthesis
Support	
Release Notes and Known Issues	Master Answer Record: 54682
All Vivado IP Change Logs	Master Vivado IP Change Logs: 72775
Support web page	

Notes:

1. For a complete list of supported devices, see the AMD Vivado™ IP catalog.
2. Standalone driver details can be found in <install_directory>/Vitis/<release>/data/embeddedsw/doc/xilinx_drivers_api_toc.htm.
3. For the supported versions of third-party tools, see the *Vivado Design Suite User Guide: Release Notes, Installation, and Licensing* (UG973).

Chapter 2

Overview

Navigating Content by Design Process

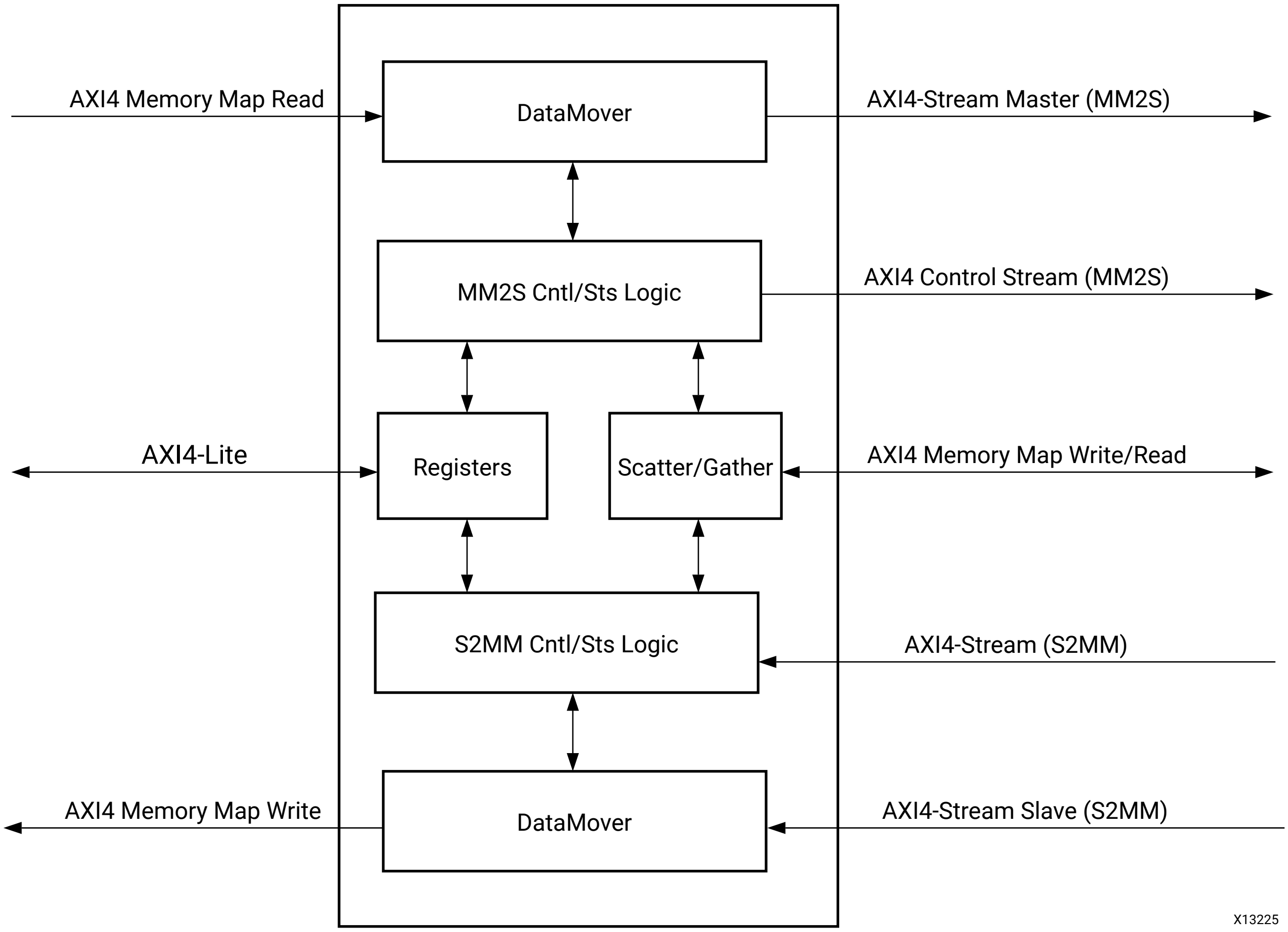
AMD Adaptive Computing documentation is organized around a set of standard design processes to help you find relevant content for your current development task. You can access the AMD Versal™ adaptive SoC design processes on the Design Hubs page. You can also use the Design Flow Assistant to better understand the design flows and find content that is specific to your intended design needs. This document covers the following design processes:

- **Hardware, IP, and Platform Development:** Creating the PL IP blocks for the hardware platform, creating PL kernels, functional simulation, and evaluating the AMD Vivado™ timing, resource use, and power closure. Also involves developing the hardware platform for system integration. Topics in this document that apply to this design process include:
    - Port Descriptions
    - Clocking
    - Resets
    - Customizing and Generating the Core
    - Chapter 6: Example Design

Core Overview

The AXI Direct Memory Access (AXI DMA) IP core provides high-bandwidth direct memory access between the AXI4 and AXI4-Stream IP interfaces. Its optional scatter gather capabilities also offload data movement tasks from the CPU in processor-based systems. Initialization, status, and management registers are accessed through an AXI4-Lite slave interface. The following figure illustrates the functional composition of the core.

Figure 1: **AXI DMA Block Diagram**



Primary high-speed DMA data movement between system memory and stream target is through the AXI4 Read Master to AXI4 memory-mapped to stream (MM2S) Master, and AXI4-Stream to memory-mapped (S2MM) Slave to AXI4 Write Master. AXI DMA also enables up to 16 multiple channels of data movement on both MM2S and S2MM paths in scatter/gather mode.

The MM2S channel and S2MM channel operate independently. The AXI DMA provides 4 KB address boundary protection (when configured in non Micro DMA), automatic burst mapping, as well as providing the ability to queue multiple transfer requests using nearly the full bandwidth capabilities of the AXI4-Stream buses. Furthermore, the AXI DMA provides byte-level data realignment allowing memory reads and writes starting at any byte offset location.

The MM2S channel supports an AXI Control stream for sending user application data to the target IP. For the S2MM channel, an AXI Status stream is provided for receiving user application data from the target IP.

The optional Scatter/Gather Engine fetches and updates buffer descriptors from system memory through the AXI4 Scatter Gather Read/Write Master interface.

Feature Summary

- AXI4 compliant
- Optional Independent Scatter/Gather Direct Memory Access (DMA) support
    - Provides offloading of DMA management work from the CPU
    - Provides fetch and update of transfer descriptors independent from primary data bus
    - Allows descriptor placement to be in any memory-mapped location separate from data buffers. For example, descriptors can be placed in block RAM.
    - Provides optional cyclic operation
- Optional Direct Register Mode (no scatter/gather support)

  A lower performance but less FPGA-resource-intensive mode can be enabled by excluding the Scatter Gather engine. In this mode transfers are commanded by setting a Source Address (for MM2S) or Destination Address (For S2MM) and then specifying a byte count in a length register.
- Primary AXI4 data width support of 32, 64, 128, 256, 512, and, 1,024 bits
- Primary AXI4-Stream data width support of 8, 16, 32, 64, 128, 256, 512, and, 1,024 bits
- Optional Data Re-alignment Engine for a stream data width up to 512 bits

  Allows data realignment to the byte (8 bits) level on the primary memory map and stream datapaths
- Optional AXI Control and Status Streams to interface to AXI Ethernet IP

  Provides optional Control Stream for the MM2S Channel and Status Stream for the S2MM channel to offload low-bandwidth control and status from the high-bandwidth datapath.
- Optional Micro mode

  AXI DMA can be configured to deliver a low footprint, low performance IP that can handle the transfer of small packets. Read the following chapters for more information.

Applications

The AXI DMA provides high-speed data movement between system memory and an AXI4-Stream-based target IP such as AXI Ethernet.

Licensing and Ordering

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

Information about other AMD LogiCORE™ IP modules is available at the Intellectual Property page. For information about pricing and availability of other AMD LogiCORE IP modules and tools, contact your local sales representative.

Chapter 3

Product Specification

Performance

Latency and Throughput

The following tables describe the latency and throughput for the AXI DMA. The tables provide performance information for a typical configuration. The throughput test consisted of transferring 10,000 bytes on the MM2S and S2MM side.

Throughput is measured from completion of descriptor fetching (DMACR.Idle = 1) to frame count interrupt assertion.

Table 1: **AXI DMA Latency Numbers**

Description	Clocks
MM2S Channel	
Tail Descriptor write to m_axi_sg_arvalid	10
m_axi_sg_arvalid to m_axi_mm2s_arvalid	28
m_axi_mm2s_arvalidto m_axis_mm2s_tvalid	6
S2MM Channel	
Tail Descriptor write to m_axi_sg_arvalid	10
s_axis_s2mm_tvalid to m_axi_s2mm_awvalid	39

Table 2: **AXI DMA Throughput Numbers**[1]

Channel	Clock Frequency (MHz)	Bytes Transferred	Total Throughput (MB/s)	Percent of Theoretical
MM2S[2]	100	10,000	399.04	99.76
S2MM[3]	100	10,000	298.59	74.64

Notes:

1. The preceding figures are measured with the default IP configuration.
2. The MM2S throughput is measured between the first `arvalid` on Memory Map side to the `tlast` on streaming side.
3. The S2MM throughput is measured between the first `tvalid` on streaming side to last `wlast` on the Memory Map side.

Resource Use

For full details about performance and resource use, visit the Performance and Resource Use web page.

Port Descriptions

The AXI DMA I/O signals are described in the following table.

Table 3: **I/O Signal Descriptions**

Signal Name	Interface	Signal Type	Init Status	Description
s_axi_lite_aclk	Clock	I		AXI4-Lite Clock.
m_axi_sg_aclk	Clock	I		AXI DMA Scatter Gather Clock
m_axi_mm2s_aclk	Clock	I		AXI DMA MM2S Primary Clock
m_axi_s2mm_aclk	Clock	I		AXI DMA S2MM Primary Clock
axi_resetn	Reset	I		AXI DMA Reset. Active-Low reset. When asserted Low, resets entire AXI DMA core. Must be synchronous to s_axi_lite_aclk.
mm2s_introut	Interrupt	O	0	Interrupt Out for Memory Map to Stream Channel.
s2mm_introut	Interrupt	O	0	Interrupt Out for Stream to Memory Map Channel.
axi_dma_tstvec	NA	O	0	Debug signals for internal use.
AXI4-Lite Interface Signals				
s_axi_lite_*	S_AXI_LITE	I/O		See Appendix A of the *Vivado Design Suite: AXI Reference Guide* (UG1037) for the AXI4 signal.
MM2S Memory Map Read Interface Signals				
m_axi_mm2s_*	M_AXI_MM2S	I/O		See Appendix A of the *Vivado Design Suite: AXI Reference Guide* (UG1037) for the AXI4 signal.
MM2S Master Stream Interface Signals				
mm2s_prmry_reset_out_n	M_AXIS_MM2S	O	1	Primary MM2S Reset Out. Active-Low reset.
m_axis_mm2s_*	M_AXIS_MM2S	I/O		See Appendix A of the *Vivado Design Suite: AXI Reference Guide* (UG1037) for the AXI4 signal.
MM2S Master Control Stream Interface Signals				
mm2s_cntrl_reset_out_n	M_AXIS_CNTRL	O	1	Control Reset Out. Active-Low reset.
m_axis_mm2s_cntrl_*	M_AXIS_CNTRL	I/O		See Appendix A of the *Vivado Design Suite: AXI Reference Guide* (UG1037) for the AXI4 signal.
S2MM Memory Map Write Interface Signals				
m_axi_s2mm_*	M_AXI_S2MM	I/O		See Appendix A of the *Vivado Design Suite: AXI Reference Guide* (UG1037) for the AXI4 signal.

Table 3: **I/O Signal Descriptions** *(cont'd)*

Signal Name	Interface	Signal Type	Init Status	Description
S2MM Slave Stream Interface Signals				
s2mm_prmry_reset_out_n	S_AXIS_S2MM	O	1	Primary S2MM Reset Out. Active-Low reset.
s_axis_s2mm_*	S_AXIS_S2MM	I	Input/ Output	See Appendix A of the *Vivado Design Suite: AXI Reference Guide* (UG1037) for the AXI4 signal.
Scatter Gather Memory Map Read Interface Signals				
m_axi_sg_*	M_AXI_SG	I/O		See Appendix A of the *Vivado Design Suite: AXI Reference Guide* (UG1037) for the AXI4 signal.
Scatter Gather Memory Map Write Interface Signals				
m_axi_sg*	M_AXI_SG	I/O		See Appendix A of the *Vivado Design Suite: AXI Reference Guide* (UG1037) for the AXI4 signal.

Register Space

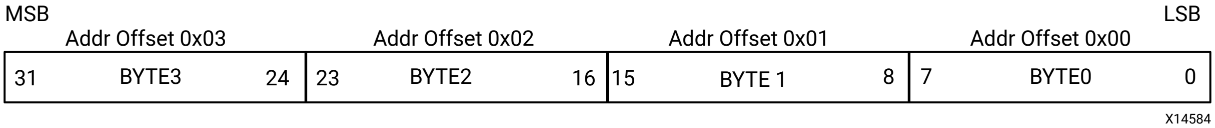
The AXI DMA core register space for Scatter/Gather Mode is shown in Table 4: Scatter/Gather Mode Register Address Map. The AXI DMA core register space for Direct Register mode is shown in Table 5: Direct Register Mode Register Address Map. The AXI DMA registers are memory-mapped into non-cacheable memory space. This memory space must be aligned on an AXI word (32-bit) boundary.

***Note*:** The AXI4-Lite write access register is updated by the 32-bit AXI Write Data (`*_wdata`) signal, and is not impacted by the AXI Write Data Strobe (`*_wstrb`) signal. For a Write, both the AXI Write Address Valid `(*_awvalid`) and AXI Write Data Valid `(*_wvalid`) signals should be asserted together.

Endianess

All registers are in Little Endian format, as shown in the following figure.

Figure 2: **32-bit Little Endian Example**



AXI DMA Register Address Map

Scatter/Gather Mode Register Address Map

Table 4: **Scatter/Gather Mode Register Address Map**

Address Space Offset[1]	Name	Description
00h	MM2S_DMACR	MM2SDMA Control register
04h	MM2S_DMASR	MM2SDMA Status register
08h	MM2S_CURDESC	MM2S Current Descriptor Pointer. Lower 32 bits of the address.
0Ch	MM2S_CURDESC_MSB	MM2S Current Descriptor Pointer. Upper 32 bits of address.
10h	MM2S_TAILDESC	MM2S Tail Descriptor Pointer. Lower 32 bits.
14h	MM2S_TAILDESC_MSB	MM2S Tail Descriptor Pointer. Upper 32 bits of address.
2Ch[2]	SG_CTL	Scatter/Gather User and Cache
30h	S2MM_DMACR	S2MM DMA Control register
34h	S2MM_DMASR	S2MM DMA Status register
38h	S2MM_CURDESC	S2MM Current Descriptor Pointer. Lower 32 address bits
3Ch	S2MM_CURDESC_MSB	S2MM Current Descriptor Pointer. Upper 32 address bits.
40h	S2MM_TAILDESC	S2MM Tail Descriptor Pointer. Lower 32 address bits.
44h	S2MM_TAILDESC_MSB	S2MM Tail Descriptor Pointer. Upper 32 address bits.

Notes:

1. Address Space Offset is relative to C_BASEADDR assignment.
2. Register 2Ch is available only when DMA is configured in multichannel mode.

Direct Register Mode Register Address Map

Table 5: **Direct Register Mode Register Address Map**

Address Space Offset[1]	Name	Description
00h	MM2S_DMACR	MM2SDMA Control register
04h	MM2S_DMASR	MM2SDMA Status register
08h– 14h	Reserved	N/A
18h	MM2S_SA	MM2S Source Address. Lower 32 bits of address.
1Ch	MM2S_SA_MSB	MM2S Source Address. Upper 32 bits of address.
28h	MM2S_LENGTH	MM2STransfer Length (Bytes)
30h	S2MM_DMACR	S2MM DMA Control register
34h	S2MM_DMASR	S2MM DMA Status register
38h– 44h	Reserved	N/A
48h	S2MM_DA	S2MM Destination Address. Lower 32 bit address.
4Ch	S2MM_DA_MSB	S2MM Destination Address. Upper 32 bit address.

Table 5: **Direct Register Mode Register Address Map** *(cont'd)*

Address Space Offset[1]	Name	Description
58h	S2MM_LENGTH	S2MMBuffer Length (Bytes)

Notes:

1. Address Space Offset is relative to C_BASEADDR assignment.

Memory Map to Stream Register Detail

Register Access Type Description

- RO = Read Only. Writing has no effect
- R/W = Read and Write Accessible
- R/WC = Read/Write to Clear

MM2S_DMACR (MM2S DMA Control Register – Offset 00h)

This register provides control for the Memory Map to Stream DMA Channel.

Figure 3: **MM2S DMACR Register**

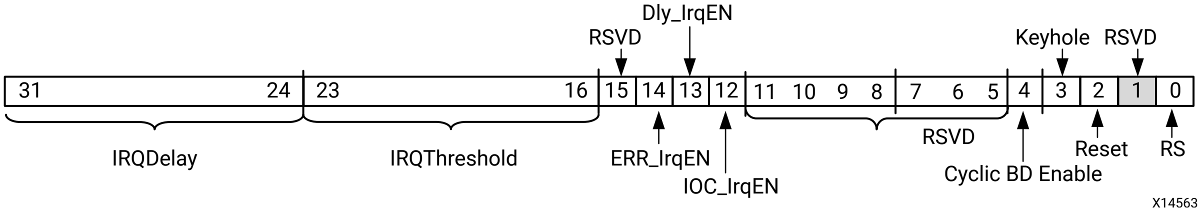


Table 6: **MM2S_DMACR Register Details**

Bits	Field Name	Default Value	Access Type	Description
0	RS	0	R/W	Run/Stop control for controlling running and stopping of the DMA channel. • 0 = Stop – DMA stops when current (if any) DMA operations are complete. For Scatter/Gather Mode pending commands/transfers are flushed or completed. AXI4-Stream outs are potentially terminated early. Descriptors in the update queue are allowed to finish updating to remote memory before engine halt. For Direct Register mode pending commands/transfers are flushed or completed. AXI4-Stream outs are potentially terminated. The halted bit in the DMA Status register asserts to 1 when the DMA engine is halted. This bit is cleared by AXI DMA hardware when an error occurs. The CPU can also choose to clear this bit to stop DMA operations. • 1 = Run – Start DMA operations. The halted bit in the DMA Status register deasserts to 0 when the DMA engine begins operations.
1	Reserved	1	RO	Writing to this bit has no effect and is always read as 1.
2	Reset	0	R/W	Soft reset for resetting the AXI DMA core. Setting this bit to a 1 causes the AXI DMA to be reset. Reset is accomplished gracefully. Pending commands/transfers are flushed or completed. AXI4-Stream outs are potentially terminated early. Setting either MM2S_DMACR.Reset = 1 or S2MM_DMACR. Reset = 1 resets the entire AXI DMA engine. After completion of a soft reset, all registers and bits are in the Reset State. • 0 = Normal operation. • 1 = Reset in progress.
3	Keyhole	0	R/W	Keyhole Read. Setting this bit to 1 causes AXI DMA to initiate MM2S reads (AXI4 read) in non-incrementing address mode (Fixed Address Burst transfer on AXI4). This bit can be updated when AXI DMA is in idle. When using keyhole operation the Max Burst Length should not exceed 16. This bit should not be set when DRE is enabled. This bit is non functional when the multichannel feature is enabled or in Direct Register mode.
4	Cyclic BD Enable	0	R/W	When set to 1, the DMA operates in Cyclic Buffer Descriptor (BD) mode without any user intervention. In this mode, the Scatter Gather module ignores the 'Completed' bit of the BD. With this bit set, you can use the same BDs in cyclic manner without worrying about any stale descriptor errors. This bit should be set/unset only when the DMA is idle or when not running. Updating this bit while the DMA is running can result in unexpected behavior. This bit is non functional when DMA operates in multichannel mode.
11 to 5	Reserved	0	RO	Writing to these bits has no effect, and they are always read as zeros.
1	IOC_IrqEn	0	R/W	Interrupt on Complete (IOC) Interrupt Enable. When set to 1, allows DMASR.IOC_Irq to generate an interrupt out for descriptors with the IOC bit set. • 0 = IOC Interrupt disabled • 1 = IOC Interrupt enabled

Table 6: **MM2S_DMACR Register Details** *(cont'd)*

Bits	Field Name	Default Value	Access Type	Description
13	Dly_IrqEn	0	R/W	Interrupt on Delay Timer Interrupt Enable. When set to 1, allows DMASR.Dly_Irq to generate an interrupt out. • 0 = Delay Interrupt disabled • 1 = Delay Interrupt enabled ***Note*: This bit is ignored when AXI DMA is configured for Direct Register Mode.
14	Err_IrqEn	0	R/W	Interrupt on Error Interrupt Enable. • 0 = Error Interrupt disabled • 1 = Error Interrupt enabled
15	Reserved	0	RO	Writing to this bit has no effect and it is always read as zeros.
23 to 16	IRQ Threshold	01h	R/W	Interrupt Threshold. This value is used for setting the interrupt threshold. When IOC interrupt events occur, an internal counter counts down from the Interrupt Threshold setting. When the count reaches zero, an interrupt out is generated by the DMA engine. ***Note***: The minimum setting for the threshold is 0x01. A write of 0x00 to this register has no effect. ***Note***: This field is ignored when AXI DMA is configured for Direct Register Mode.
31 to 24	IRQ Delay	00h	R/W	Interrupt Delay Time Out. This value is used for setting the interrupt timeout value. The interrupt timeout mechanism causes the DMA engine to generate an interrupt after the delay time period has expired. Timer begins counting at the end of a packet and resets with receipt of a new packet or a timeout event occurs. 1 Timeout Interval = 125´ (clock period of SG clock) Setting a value of 3 here results in a delay timeout of 125 x 3 x (clock period of SG clock). ***Note***: Setting this value to zero disables the delay timer interrupt. ***Note***: This field is ignored when AXI DMA is configured for Direct Register Mode.

MM2S_DMASR (MM2S DMA Status Register – Offset 04h)

This register provides the status for the Memory Map to Stream DMA Channel.

Figure 4: **MM2S DMASR Register**

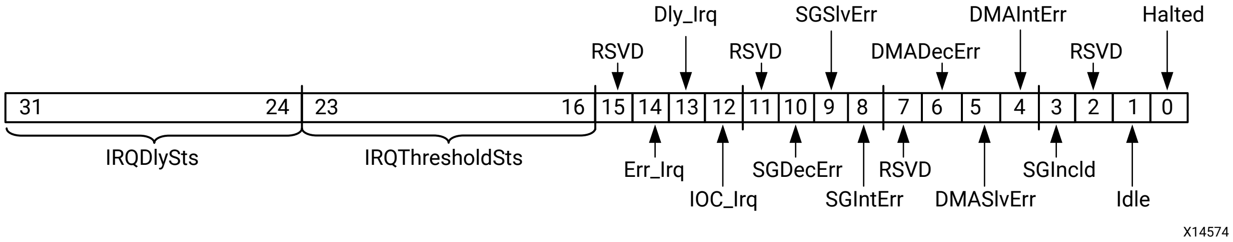


Table 7: **MM2S_DMASR Register Details**

Bits	Field Name	Default Value	Access Type	Description
0	Halted	1	RO	DMA Channel Halted. Indicates the run/stop state of the DMA channel. • 0 = DMA channel running. • 1 = DMA channel halted. For Scatter / Gather Mode this bit gets set when DMACR.RS = 0 and DMA and Scatter Gather (SG) operations have halted. For Direct Register mode (C_INCLUDE_SG = 0) this bit gets set when DMACR.RS = 0 and DMA operations have halted. There can be a lag of time between when DMACR.RS = 0 and when DMASR.Halted = 1. ***Note***: When halted (RS= 0 and Halted = 1), writing to TAILDESC_PTR pointer registers has no effect on DMA operations when in Scatter Gather Mode. For Direct Register Mode, writing to the LENGTH register has no effect on DMA operations.
1	Idle	0	RO	DMA Channel Idle. Indicates the state of AXI DMA operations. For Scatter/Gather Mode when IDLE indicates the SG Engine has reached the tail pointer for the associated channel and all queued descriptors have been processed. Writing to the tail pointer register automatically restarts DMA operations. The IDLE bit is associated with the BDs. The DMA might be in IDLE state, there might be active data on the AXI interface. For Direct Register Mode when IDLE indicates the current transfer has completed. • 0 = Not Idle. For Scatter/Gather Mode, SG has not reached tail descriptor pointer and/or DMA operations in progress. For Direct Register Mode, transfer is not complete. • 1 = Idle. For Scatter/Gather Mode, SG has reached tail descriptor pointer and DMA operation paused. for Direct Register Mode, DMA transfer has completed and controller is paused. ***Note***: This bit is 0 when channel is halted (DMASR.Halted=1). This bit is also 0 prior to initial transfer when AXI DMA configured for Direct Register Mode.
2	Reserved	0	RO	Writing to this bit has no effect and it is always read as zero.
3	SGIncld	C_ INCLUDE_ SG	RO	• 1= Scatter Gather Enabled • 0= Scatter Gather not enabled
4	DMAIntErr	0	RO	DMA Internal Error. Internal error occurs if the buffer length specified in the fetched descriptor is set to 0. This error condition causes the AXI DMA to halt gracefully. The DMACR.RS bit is set to 0, and when the engine has completely shut down, the DMASR.Halted bit is set to 1. • 0 = No DMA Internal Errors • 1 = DMA Internal Error detected. DMA Engine halts. ***Note***: This bit is not used and is fixed at 0 when AXI DMA is configured for Direct Register Mode.

Table 7: **MM2S_DMASR Register Details** *(cont'd)*

Bits	Field Name	Default Value	Access Type	Description
5	DMASlvErr	0	RO	DMA Slave Error. This error occurs if the slave read from the Memory Map interface issues a Slave Error. This error condition causes the AXI DMA to halt gracefully. The DMACR.RS bit is set to 0, and when the engine has completely shut down, the DMASR.Halted bit is set to 1. • 0 = No DMA Slave Errors. • 1 = DMA Slave Error detected. DMA Engine halts.
6	DMADecErr	0	RO	DMA Decode Error. This error occurs if the address request points to an invalid address. This error condition causes the AXI DMA to halt gracefully. The DMACR.RS bit is set to 0, and when the engine has completely shut down, the DMASR.Halted bit is set to 1. • 0 = No DMA Decode Errors. • 1 = DMA Decode Error detected. DMA Engine halts.
7	Reserved	0	RO	Writing to this bit has no effect, and it is always read as zeros.
8	SGIntErr	0	RO	Scatter Gather Internal Error. This error occurs if a descriptor with the “Complete bit” already set is fetched. Refer to the Scatter Gather Descriptor section for more information. This indicates to the SG Engine that the descriptor is a stale descriptor. This error condition causes the AXI DMA to halt gracefully. The DMACR.RS bit is set to 0, and when the engine has completely shut down, the DMASR.Halted bit is set to 1. • 0 = No SG Internal Errors. • 1 = SG Internal Error detected. DMA Engine halts. ***Note*:** This bit is not used and is fixed at 0 when AXI DMA is configured for Direct Register Mode.
9	SGSlvErr	0	RO	Scatter Gather Slave Error. This error occurs if the slave read from on the Memory Map interface issues a Slave error. This error condition causes the AXI DMA to halt gracefully. The DMACR.RS bit is set to 0, and when the engine has completely shut down, the DMASR.Halted bit is set to 1. • 0 = No SG Slave Errors. • 1 = SG Slave Error detected. DMA Engine halts. ***Note*:** This bit is not used and is fixed at 0 when AXI DMA is configured for Direct Register Mode.
10	SGDecErr	0	RO	Scatter Gather Decode Error. This error occurs if CURDESC_PTR and/or NXTDESC_PTR points to an invalid address. This error condition causes the AXI DMA to halt gracefully. The DMACR.RS bit is set to 0, and when the engine has completely shut down, the DMASR.Halted bit is set to 1. • 0 = No SG Decode Errors. • 1 = SG Decode Error detected. DMA Engine halts. ***Note*:** This bit is not used and is fixed at 0 when AXI DMA is configured for Direct Register Mode.
11	Reserved	0	RO	Writing to this bit has no effect, and it is always read as zeros.

Table 7: **MM2S_DMASR Register Details** *(cont'd)*

Bits	Field Name	Default Value	Access Type	Description
12	IOC_Irq	0	R/WC	Interrupt on Complete. When set to 1 for Scatter/Gather Mode, indicates an interrupt event was generated on completion of a descriptor. This occurs for descriptors with the End of Frame (EOF) bit set. When set to 1 for Direct Register Mode, indicates an interrupt event was generated on completion of a transfer. If the corresponding bit is enabled in the MM2S_DMACR (IOC_IrqEn = 1) and if the interrupt threshold has been met, causes an interrupt out to be generated from the AXI DMA. • 0 = No IOC Interrupt. • 1 = IOC Interrupt detected. Writing a 1 to this bit clears it.
13	Dly_Irq	0	R/WC	Interrupt on Delay. When set to 1, indicates an interrupt event was generated on delay timer timeout. If the corresponding bit is enabled in the MM2S_DMACR (Dly_IrqEn = 1), an interrupt out is generated from the AXI DMA. • 0 = No Delay Interrupt. • 1 = Delay Interrupt detected.1 = IOC Interrupt detected. Writing a 1 to this bit clears it. ***Note*: This bit is not used and is fixed at 0 when AXI DMA is configured for Direct Register Mode.**
14	Err_Irq	0	R/WC	Interrupt on Error. When set to 1, indicates an interrupt event was generated on error. If the corresponding bit is enabled in the MM2S_DMACR (Err_IrqEn = 1), an interrupt out is generated from the AXI DMA. Writing a 1 to this bit clears it. • 0 = No error Interrupt. • 1 = Error interrupt detected.
15	Reserved	0	RO	Always read as zero.
23 to 16	IRQThresholdSts	01h	RO	Interrupt Threshold Status. Indicates current interrupt threshold value. The value programmed in the IRQ Threshold field in MM2S_CR is decremented on every packet transfer and reflected here. Before the DMA is started or before sending the first packet, this register will have the same value as programmed in the IRQ Threshold field of MM2S_CR. ***Note***: Applicable only when Scatter Gather is enabled.
31 to 24	IRQDelaySts	00h	RO	Interrupt Delay Time Status. Indicates current interrupt delay time value. ***Note***: Applicable only when Scatter Gather is enabled.

MM2S_CURDESC (MM2S DMA Current Descriptor Pointer Register - Offset 08h)

This register provides the Current Descriptor Pointer for the Memory Map to Stream DMA Scatter Gather Descriptor Management.

Figure 5: **MM2S CURDESC Register**

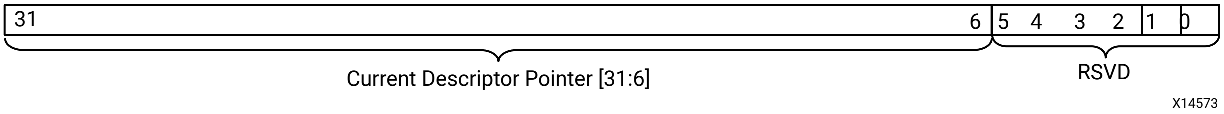


Table 8: **MM2S_CURDESC Register Details**

Bits	Field Name	Default Value	Access Type	Description
5 to 0 (Offset 0x38)	Reserved	0	RO	Writing to these bits has no effect and they are always read as zeros.
31 to 6	Current Descriptor Pointer	zeros	R/W (RO)	Indicates the pointer of the current descriptor being worked on. This register must contain a pointer to a valid descriptor prior to writing the TAILDESC_PTR register. Otherwise, undefined results occur. When DMACR.RS is 1, CURDESC_PTR becomes Read Only (RO) and is used to fetch the first descriptor. When the DMA Engine is running (DMACR.RS=1),CURDESC_PTR registers are updated by AXI DMA to indicate the current descriptor being worked on. On error detection, CURDESC_PTR is updated to reflect the descriptor associated with the detected error. ***Note*:** The register can only be written to by the CPU when the DMA Engine is Halted (DMACR.RS=0 and DMASR.Halted =1). At all other times, this register is Read Only (RO). Descriptors must be 16 word aligned, that is, 0x00, 0x40, 0x80 and others. Any other alignment has undefined results.

MM2S_CURDES_MSB (MM2S DMA Current Descriptor Pointer Register - Offset 0Ch)

This register provides the upper 32 bits of Current Descriptor Pointer for the Memory Map to Stream DMA Scatter Gather Descriptor Management. This is applicable only when address space is more than 32 bits.

Figure 6: **MM2S CURDESC_MSB Register**

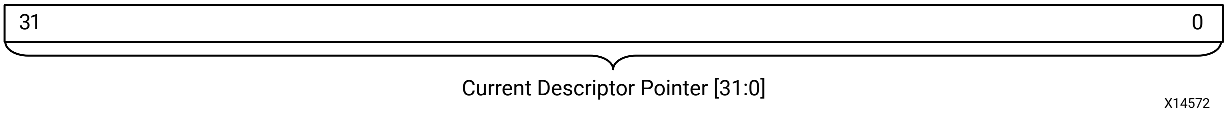


Table 9: **MM2S_CURDESC_MSB Register Details**

Bits	Field Name	Default Value	Access Type	Description
31 to 0	Current Descriptor Pointer	zeros	R/W (RO)	Indicates the pointer of the current descriptor being worked on. This register must contain a pointer to a valid descriptor prior to writing the TAILDESC_PTR register. Otherwise, undefined results occur. When DMACR.RS is 1, CURDESC_PTR becomes Read Only (RO) and is used to fetch the first descriptor. When the DMA Engine is running (DMACR.RS=1),CURDESC_PTR registers are updated by AXI DMA to indicate the current descriptor being worked on. On error detection, CURDESC_PTR is updated to reflect the descriptor associated with the detected error. ***Note***: The register can only be written to by the CPU when the DMA Engine is Halted (DMACR.RS=0 and DMASR.Halted =1). At all other times, this register is Read Only (RO). Descriptors must be 16 word aligned, that is, 0x00, 0x40, 0x80 and others. Any other alignment has undefined results.

MM2S_TAILDESC (MM2S DMA Tail Descriptor Pointer Register - Offset 10h)

This register provides the Tail Descriptor Pointer for the Memory Map to Stream DMA Scatter Gather Descriptor Management.

Figure 7: **MM2S_TAILDESC Register**

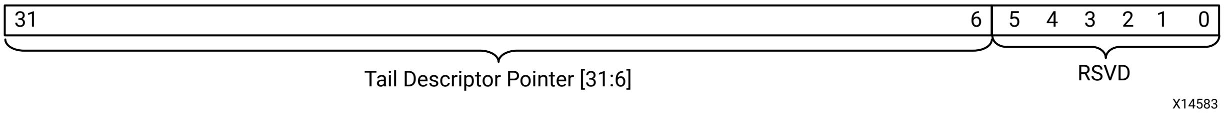


Table 10: **MM2S_TAILDESC Register Details**

Bits	Field Name	Default Value	Access Type	Description
5 to 0	Reserved	0	RO	Writing to these bits has no effect, and they are always read as zeros.

Table 10: **MM2S_TAILDESC Register Details** *(cont'd)*

Bits	Field Name	Default Value	Access Type	Description
31 to 6	Tail Descriptor Pointer	zeros	R/W	Indicates the pause pointer in a descriptor chain. The AXI DMA SG Engine pauses descriptor fetching after completing operations on the descriptor whose current descriptor pointer matches the tail descriptor pointer. When AXI DMA Channel is not halted (DMASR.Halted = 0), a write by the CPU to the TAILDESC_PTR register causes the AXI DMASG Engine to start fetching descriptors or restart if it was idle (DMASR.Idle = 1). If it was not idle, writing TAILDESC_PTR has no effect except to reposition the pause point. ***Note***: The software must not move the tail pointer to a location that has not been updated. The software processes and reallocates all completed descriptors (Cmplted = 1), clears the completed bits and then moves the tail pointer. The software must move the pointer to the last descriptor it updated. Descriptors must be 16-word aligned, that is, 0x00, 0x40, 0x80, and so forth. Any other alignment has undefined results.

MM2S_TAILDESC_MSB (MM2S DMA Tail Descriptor Pointer Register – Offset 14h)

This register provides the upper 32 bits of Tail Descriptor Pointer for the Memory Map to Stream DMA Scatter Gather Descriptor Management. This is applicable only when the address space is more than 32 bits wide.

Figure 8: **MM2S_TAILDESC_MSB Register**

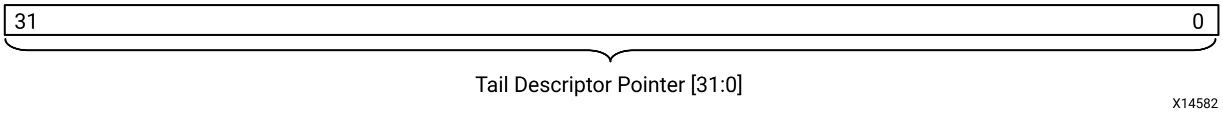


Table 11: **MM2S_TAILDESC_MSB Register Details**

Bits	Field Name	Default Value	Access Type	Description
31 to 0	Tail Descriptor Pointer	zeros	R/W	Indicates the pause pointer in a descriptor chain. The AXI DMA SG Engine pauses descriptor fetching after completing operations on the descriptor whose current descriptor pointer matches the tail descriptor pointer. When AXI DMA Channel is not halted (DMASR.Halted = 0), a write by the CPU to the TAILDESC_PTR_MSB register causes the AXI DMA SG Engine to start fetching descriptors or restart if it was idle (DMASR.Idle = 1). If it was not idle, writing TAILDESC_PTR has no effect except to reposition the pause point. If the AXI DMA Channel is halted (DMASR.Halted = 1 and DMACR.RS = 0), a write by the CPU to the TAILDESC_PTR register has no effect except to reposition the pause point. ***Note*****:** The software must not move the tail pointer to a location that has not been updated. The software processes and reallocates all completed descriptors (Cmplted = 1), clears the completed bits and then moves the tail pointer. The software must move the pointer to the last descriptor it updated. Descriptors must be 16-word aligned, that is, 0x00, 0x40, 0x80, and so forth. Any other alignment has undefined results.

MM2S_SA (MM2S DMA Source Address Register – Offset 18h)

This register provides the Source Address for reading system memory for the Memory Map to Stream DMA transfer.

Figure 9: **MM2S_SA Register**

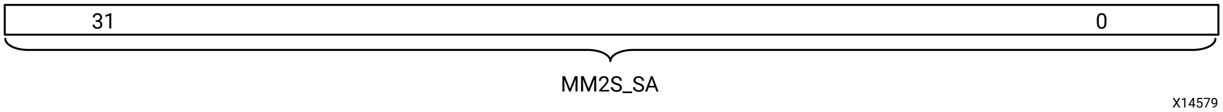


Table 12: **MM2S_SA Register Details**

Bits	Field Name	Default Value	Access Type	Description
31 to 0	Source Address	zeros	R/W	Indicates the source address AXI DMA reads from to transfer data to AXI4-Stream on the MM2S Channel. ***Note*****:** If Data Realignment Engine is included, the Source Address can be at any byte offset. If Data Realignment Engine is not included, the Source Address must be MM2S Memory Map data width aligned.

MM2S_SA_MSB (MM2S DMA Source Address Register – Offset 1Ch)

This register provides the upper 32 bits of the Source Address for reading system memory for the Memory Map to Stream DMA transfer. This is applicable only when the DMA is configured for an address space greater than 32.

Figure 10: **MM2S_SA_MSB Register**

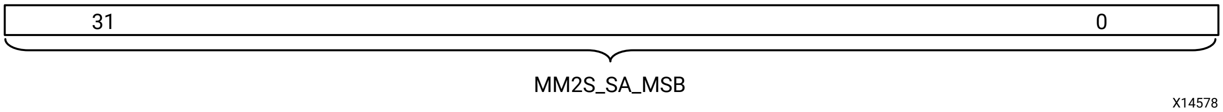


Table 13: **MM2S_SA_MSB Register Details**

Bits	Field Name	Default Type	Access Type	Description
31 to 0	Source Address	zeros	R/W	Indicates the MSB 32 bits of the source address AXI DMA reads from to transfer data to AXI4-Stream on the MM2S Channel. ***Note***: If Data Realignment Engine is included, the Source Address can be at any byte offset. If Data Realignment Engine is not included, the Source Address must be MM2S Memory Map data width aligned.

MM2S_LENGTH (MM2S DMA Transfer Length Register — Offset 28h)

This register provides the number of bytes to read from system memory and transfer to MM2S AXI4-Stream.

Figure 11: **MM2S_LENGTH Register**

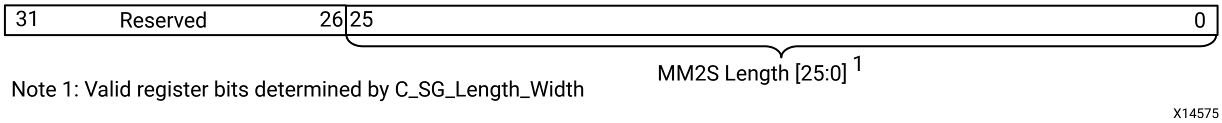


Table 14: **MM2S_LENGTH Register Details**

Bits	Field Name	Default Value	Access Type	Description
25[1] to 0	Length	zeros	R/W	Indicates the number of bytes to transfer for the MM2S channel. Writing a non-zero value to this register starts the MM2S transfer.
31 to 26	Reserved	0	RO	Writing to these bits has no effect and they are always read as zeros.

Notes:

1. Width of Length field determined by Buffer Length Register Width parameter. Minimum width is 8 bits (7 to 0) and maximum width is 26 bits (25 to 0).

SG_CTL (Scatter/Gather User and Cache Control Register—Offset 2Ch)

This register is available only when DMA is configured in multichannel mode.

Figure 12: **SG_CTL Register**

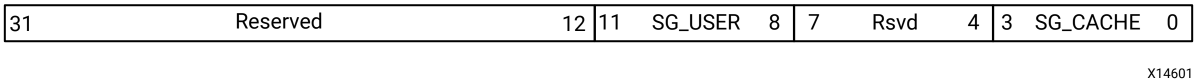


Table 15: **SG_CTL Register Details**

Bits	Field Name	Default Value	Access Type	Description
3 to 0	SG_CACHE	0011b	R/W	Scatter/Gather Cache Control. Values written in this register reflect on the m_axi_sg_arcache and m_axi_sg_awcache signals of the M_AXI_SG interface.
7 to 4	Reserved	0	RO	Writing to these bits has no effect and they are always read as zeros.
11 to 8	SG_USER	0	R/W	Scatter/Gather User Control. Values written in this register reflect on the m_axi_sg_aruser and m_axi_sg_awuser signals of the M_AXI_SG interface.

Stream to Memory Map Register Detail

S2MM_DMACR (S2MM DMA Control Register – Offset 30h)

This register provides control for the Stream to Memory Map DMA Channel.

Figure 13: **S2MM_DMACR Register**

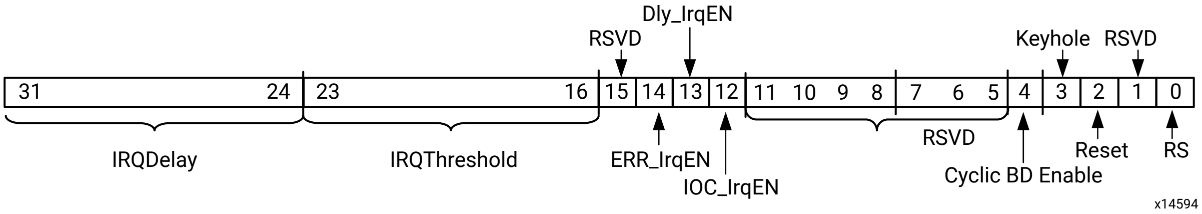


Table 16: **S2MM_DMACR Register Details**

Bits	Field Name	Default Value	Access	Description
0	RS	0	R/W	Run/Stop control for controlling running and stopping of the DMA channel. • 0 = Stop – DMA stops when current (if any) DMA operations are complete. For Scatter/Gather Mode pending commands/ transfers are flushed or completed. AXI4-Streams are potentially terminated early. Descriptors in the update queue are allowed to finish updating to remote memory before engine halt. For Direct Register Mode pending commands/transfers are flushed or completed. AXI4-Streams are potentially terminated. Data integrity on S2MM AXI4 cannot be guaranteed. The halted bit in the DMA Status register asserts to 1 when the DMA engine is halted. This bit is cleared by AXI DMA hardware when an error occurs. The CPU can also choose to clear this bit to stop DMA operations. • 1 = Run – Start DMA operations. The halted bit in the DMA Status Register deasserts to 0 when the DMA engine begins operations.
1	Reserved	1	RO	Writing to this bit has no effect, and is always read as 1.
2	Reset	0	R/W	Soft reset for resetting the AXI DMA core. Setting this bit to a 1 causes the AXI DMA to be reset. Reset is accomplished gracefully. Pending commands/transfers are flushed or completed. AXI4-Stream outs are terminated early, if necessary with associated TLAST. Setting either MM2S_DMACR.Reset = 1 or S2MM_DMACR. Reset = 1 resets the entire AXI DMA engine. After completion of a soft reset, all registers and bits are in the Reset State. • 0 = Reset not in progress. Normal operation. • 1 = Reset in progress.
3	Keyhole	0	R/W	Keyhole Write. Setting this bit to 1 causes AXI DMA to initiate S2MM writes (AXI4 Writes) in non-incrementing address mode (Fixed Address Burst transfer on AXI4). This bit can be modified when AXI DMA is in idle. When enabling Key hole operation the maximum burst length cannot be more than 16. This bit should not be set when DRE is enabled. This bit is non functional when DMA is used in multichannel mode.
4	Cyclic BD Enable	0	R/W	When set to 1, you can use the DMA in Cyclic Buffer Descriptor (BD) mode without any user intervention. In this mode, the Scatter Gather module ignores the 'Completed' bit of the BD. With this feature, you can use the same BDs in cyclic manner without worrying about any stale descriptor errors. This bit is non functional when DMA operates in MultiChannel mode or in Direct Register Mode.
11 to 5	Reserved	0	RO	Writing to these bits has no effect and they are always read as zeros.
12	IOC_IRqEn	0	R/W	Interrupt on Complete Interrupt Enable. When set to 1, allows Interrupt On Complete events to generate an interrupt out for descriptors with the "Complete bit" bit set. • 0 = IOC Interrupt disabled. • 1 = IOC Interrupt enabled.

Table 16: **S2MM_DMACR Register Details** *(cont'd)*

Bits	Field Name	Default Value	Access	Description
13	Dly_IrqEn	0	R/W	Interrupt on Delay Timer Interrupt Enable. When set to 1, allows error events to generate an interrupt out. • 0 = Delay Interrupt disabled. • 1 = Delay Interrupt enabled. ***Note***: Applicable only when Scatter Gather is enabled.
14	Err_IrqEn	0	R/W	Interrupt on Error Interrupt Enable. When set to 1, allows error events to generate an interrupt out. • 0 = Error Interrupt disabled. • 1 = Error Interrupt enabled.
15	Reserved	0	RO	Writing to this bit has no effect, and it is always read as zeros.
23 to 16	IRQThreshold	01h	R/W	Interrupt Threshold. This value is used for setting the interrupt threshold. When IOC interrupt events occur, an internal counter counts down from the Interrupt Threshold setting. When the count reaches zero, an interrupt out is generated by the DMA engine. ***Note***: The minimum setting for the threshold is 0x01. A write of 0x00 to this register has no effect. ***Note***: Applicable only when Scatter Gather is enabled.
31 to 24	IRQDelay	00h	R/W	Interrupt Delay Time Out. This value is used for setting the interrupt timeout value. The interrupt timeout is a mechanism for causing the DMA engine to generate an interrupt after the delay time period has expired. The timer begins counting at the end of a packet and resets with the receipt of a new packet or a timeout event occurs. 1 Timeout Interval =125´ (clock period of SG clock) Setting a value of 3 here results in a delay timeout of 125 x 3 x (clock period of SG clock). ***Note***: Setting this value to zero disables the delay timer interrupt. ***Note***: Applicable only when Scatter Gather is enabled.

S2MM_DMASR (S2MM DMA Status Register – Offset 34h)

This register provides the status for the Stream to Memory Map DMA Channel.

Figure 14: **S2MM DMASR Register**

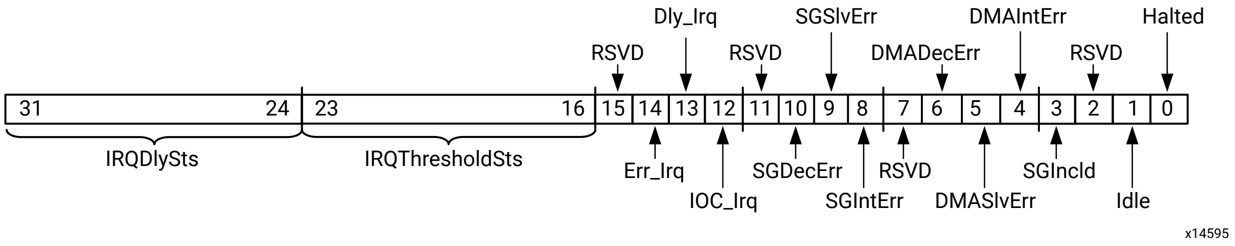


Table 17: **S2MM DMASR Register Details**

Bits	Field Name	Default Value	Access Type	Description
0	Halted	1	RO	DMA Channel Halted. Indicates the run/stop state of the DMA channel. • 0 = DMA channel running. • 1 = DMA channel halted. For Scatter/Gather Mode this bit gets set when DMACR.RS = 0 and DMA and SG operations have halted. For Direct Register Mode this bit gets set when DMACR.RS = 0 and DMA operations have halted. There can be a lag of time between when DMACR.RS = 0 and when DMASR.Halted = 1. ***Note***: When halted (RS= 0 and Halted = 1), writing to TAILDESC_PTR pointer registers has no effect on DMA operations when in Scatter Gather Mode. For Direct Register Mode, writing to the LENGTH register has no effect on DMA operations.
1	Idle	0	RO	DMA Channel Idle. Indicates the state of AXI DMA operations. For Scatter /Gather Mode when IDLE indicates the SG Engine has reached the tail pointer for the associated channel and all queued descriptors have been processed. Writing to the tail pointer register automatically restarts DMA operations. For Direct Register Mode when IDLE indicates the current transfer has completed. • 0 = Not Idle. • 1 = Idle. ***Note***: This bit is 0 when channel is halted (DMASR.Halted=1). This bit is also 0 prior to initial transfer when AXI DMA is configured for Direct Register Mode.
2	Reserved	0	RO	Writing to this bit has no effect and it is always read as zero.
3	SGIncld	C_ INCLUDE_ SG	RO	Scatter Gather Engine Included. DMASR.SGIncld = 1 indicates the Scatter Gather engine is included and the AXI DMA is configured for Scatter Gather mode. DMASR.SGIncld = 0 indicates the Scatter Gather engine is excluded and the AXI DMA is configured for Direct Register Mode.
4	DMAIntErr	0	RO	DMA Internal Error. This error occurs if the buffer length specified in the fetched descriptor is set to 0. Also, when in Scatter Gather Mode and using the status app length field, this error occurs when the Status AXI4-Stream packet RxLength field does not match the S2MM packet being received by the S_AXIS_S2MM interface. When Scatter Gather is disabled, this error is flagged if any error occurs during Memory write or if the incoming packet is bigger than what is specified in the DMA length register. This error condition causes the AXI DMA to halt gracefully. The DMACR.RS bit is set to 0, and when the engine has completely shutdown, the DMASR.Halted bit is set to 1. • 0 = No DMA Internal Errors • 1 = DMA Internal Error detected.

Table 17: **S2MM DMASR Register Details** *(cont'd)*

Bits	Field Name	Default Value	Access Type	Description
5	DMASlvErr	0	RO	DMA Slave Error. This error occurs if the slave read from the Memory Map interface issues a Slave Error. This error condition causes the AXI DMA to halt gracefully. The DMACR.RS bit is set to 0 and when the engine has completely shut down the DMASR.Halted bit is set to 1. • 0 = No DMA Slave Errors • 1 = DMA Slave Error detected.
6	DMADecErr	0	RO	DMA Decode Error. This error occurs if the address request points to an invalid address. This error condition causes the AXI DMA to halt gracefully. The DMACR.RS bit is set to 0, and when the engine has completely shut down, the DMASR.Halted bit is set to 1. • 0 = No DMA Decode Errors • 1 = DMA Decode Error detected.
7	Reserved	0	RO	Writing to this bit has no effect and it is always read as zero.
8	SGIntErr	0	RO	Scatter Gather Internal Error. This error occurs if a descriptor with the Complete bit already set is fetched. This indicates to the SG Engine that the descriptor is a tail descriptor. This error condition causes the AXI DMA to halt gracefully. The DMACR.RS bit is set to 0, and when the engine has completely shut down, the DMASR.Halted bit is set to 1. • 0 = No SG Internal Errors • 1 = SG Internal Error detected. This error cannot be logged into the descriptor. ***Note***: Applicable only when Scatter Gather is enabled
9	SGSlvErr	0	RO	Scatter Gather Slave Error. This error occurs if the slave read from on the Memory Map interface issues a Slave Error. This error condition causes the AXI DMA to gracefully halt. The DMACR.RS bit is set to 0, and when the engine has completely shut down, the DMASR.Halted bit is set to 1. • 0 = No SG Slave Errors • 1 = SG Slave Error detected. DMA Engine halts. This error cannot be logged into the descriptor. ***Note***: Applicable only when Scatter Gather is enabled.
10	SGDecErr	0	RO	Scatter Gather Decode Error. This error occurs if CURDESC_PTR and/or NXTDESC_PTR point to an invalid address. This error condition causes the AXI DMA to halt gracefully. The DMACR.RS bit is set to 0 and when the engine has completely shut down, the DMASR.Halted bit is set to 1. • 0 = No SG Decode Errors • 1 = SG Decode Error detected. DMA Engine halts. This error cannot be logged into the descriptor. ***Note***: Applicable only when Scatter Gather is enabled.
11	Reserved	0	RO	Writing to this bit has no effect and it is always read as zeros.

Table 17: **S2MM DMASR Register Details** *(cont'd)*

Bits	Field Name	Default Value	Access Type	Description
12	IOC_Irq	0	R/WC	Interrupt on Complete. When set to 1 for Scatter/Gather Mode indicates an interrupt event was generated on completion of a descriptor. This occurs for descriptors with the End of Frame (EOF) bit set. When set to 1 for Direct Register Mode indicates an interrupt event was generate on completion of a transfer. If the corresponding bit in S2MM_DMACR is enabled (IOC_IrqEn = 1) and if the interrupt threshold has been met, causes an interrupt out to be generated from the AXI DMA. • 0 = No IOC Interrupt • 1 = IOC Interrupt detected. Writing a 1 to this bit clears it.
13	Dly_Irq	0	R/WC	Interrupt on Delay. When set to 1, indicates an interrupt event was generated on delay timer timeout. If the corresponding bit in S2MM_DMACR is enabled (Dly_IrqEn = 1), an interrupt out is generated from the AXI DMA. • 0 = No Delay Interrupt • 1 = Delay Interrupt detected. Writing a 1 to this bit clears it. ***Note***: Applicable only when Scatter Gather is enabled.
14	Err_Irq	0	R/WC	Interrupt on Error. When set to 1, indicates an interrupt event was generated on error. If the corresponding bit in S2MM_DMACR is enabled (Err_IrqEn = 1), an interrupt out is generated from the AXI DMA. Writing a 1 to this bit clears it. • 0 = No Error Interrupt • 1 = Error Interrupt detected.
15	Reserved	0	RO	Writing to this bit has no effect and it is always read as zeros.
23 to 16	IRQThresholdSts	01h	RO	Interrupt Threshold Status. Indicates current interrupt threshold value. The value programmed in the IRQThreshold field in S2MM_CR is decremented on every packet transfer and reflected here. Before the DMA is started or before receiving the first packet, this register will have the same value as programmed in the IRQThreshold field of S2MM_CR. ***Note***: Applicable only when Scatter Gather is enabled.
31 to 24	IRQDelaySts	00h	RO	Interrupt delay time Status. Indicates current interrupt delay time value. ***Note***: Applicable only when Scatter Gather is enabled.

S2MM_CURDESC (S2MM DMA Current Descriptor Pointer Register – Offset 38h)

This register provides the Current Descriptor Pointer for the Stream to Memory Map DMA Scatter Gather Descriptor Management.

Figure 15: **S2MM CURDESC Register**

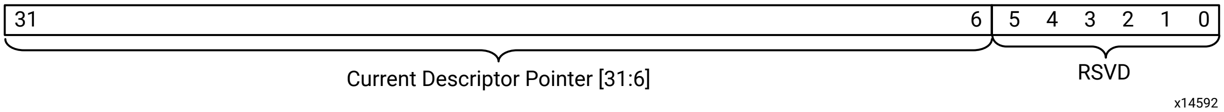


Table 18: **S2MM_CURDESC Register Details**

Bits	Field Name	Default Value	Access Type	Description
5 to 0 (Offset 0x38)	Reserved	0	RO	Writing to these bits has no effect and they are always read as zeros.
31 to 6	Current Descriptor Pointer	zeros	R/W (RO)	Indicates the pointer of the current Buffer Descriptor being worked on. This register must contain a pointer to a valid descriptor prior to writing to the TAILDESC_PTR register. Otherwise, undefined results occur. When DMACR.RS is 1, CURDESC_PTR becomes Read Only (RO) and is used to fetch the first descriptor. When the DMA Engine is running (DMACR.RS=1),CURDESC_PTR registers are updated by AXI DMA to indicate the current descriptor being worked on. On error detection, CURDESC_PTR is updated to reflect the descriptor associated with the detected error. ***Note*:** The register can only be written to by the CPU when the DMA Engine is halted (DMACR.RS=0 and DMASR.Halted =1). At all other times, this register is Read Only (RO). Buffer Descriptors must be 16-word aligned, that is, 0x00, 0x40, 0x80, and so forth. Any other alignment has undefined results.

S2MM_CURDESC_MSB (S2MM DMA Current Descriptor Pointer Register – Offset 3Ch)

This register provides the upper 32 bits of Current Descriptor Pointer for the Stream to Memory Map DMA Scatter Gather Descriptor Management. This is used only when DMA is configured for address space greater than 32 bits.

Figure 16: **S2MM CURDESC_MSB Register**

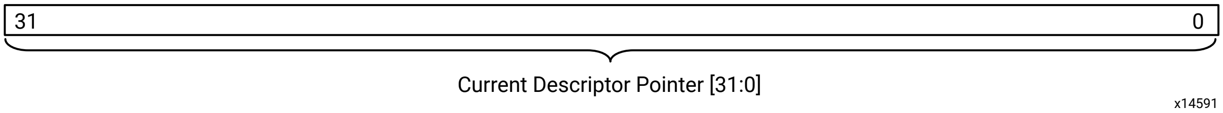


Table 19: **S2MM CURDESC_MSB Register Details**

Bits	Field Name	Default Value	Access Type	Description
31 to 0	Current Descriptor Pointer	zeros	R/W (RO)	Indicates the pointer of the current Buffer Descriptor being worked on. This register must contain a pointer to a valid descriptor prior to writing the TAILDESC_PTR register. Otherwise, undefined results occur. When DMACR.RS is 1,CURDESC_PTR becomes Read Only (RO) and is used to fetch the first descriptor. When the DMA Engine is running (DMACR.RS=1), CURDESC_PTR registers are updated by AXI DMA to indicate the current descriptor being worked on. On error detection, CURDESC_PTR is updated to reflect the descriptor associated with the detected error. ***Note*: The register can only be written to by the CPU when the DMA Engine is halted (DMACR.RS=0 and DMASR.Halted =1). At all other times, this register is Read Only (RO).

S2MM_TAILDESC (S2MM DMA Tail Descriptor Pointer Register – Offset 40h)

This register provides the Tail Descriptor Pointer for the Stream to Memory Map DMA Scatter Gather Descriptor Management.

Figure 17: **S2MM TAILDESC Register Details**

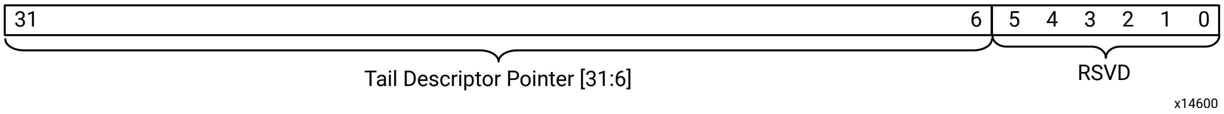


Table 20: **S2MM TAILDESC Register**

Bits	Field Name	Default Access	Access Type	Description
5 to 0	Reserved	0	RO	Writing to these bits has no effect and they are always read as zeros.

Table 20: **S2MM TAILDESC Register** *(cont'd)*

Bits	Field Name	Default Access	Access Type	Description
31 to 6	Tail Descriptor Pointer	zeros	R/W	Indicates the pause pointer in a descriptor chain. The AXI DMA SG Engine pauses descriptor fetching after completing operations on the descriptor whose current descriptor pointer matches the tail descriptor pointer. When AXI DMA Channel is not halted (DMASR.Halted = 0), a write by the CPU to the TAILDESC_PTR register causes the AXI DMA SG Engine to start fetching descriptors or restart if it was idle (DMASR.Idle = 1). If it was not idle, then writing to TAILDESC_PTR has no effect except to reposition the pause point. If the AXI DMA Channel DMACR.RS bit is set to 0 (DMASR.Halted =1 and DMACR.RS = 0), a write by the CPU to the TAILDESC_PTR register has no effect except to reposition the pause point. ***Note***: The software must not move the Tail Pointer to a location that has not been updated. The software processes and reallocates all completed descriptors (Cmplted = 1), clears the completed bits and then moves the tail pointer. The software must move the pointer to the last descriptor it updated. Descriptors must be 16-word aligned, that is, 0x00, 0x40, 0x80, and so forth. Any other alignment has undefined results.

S2MM_TAILDESC_MSB (S2MM DMA Tail Descriptor Pointer Register – Offset 44h)

This register provides the upper 32 bits of Tail Descriptor Pointer for the Stream to Memory Map DMA Scatter Gather Descriptor Management. This is used when DMA is configured for address space greater than 32.

Figure 18: **S2MM TAILDESC_MSB Register**

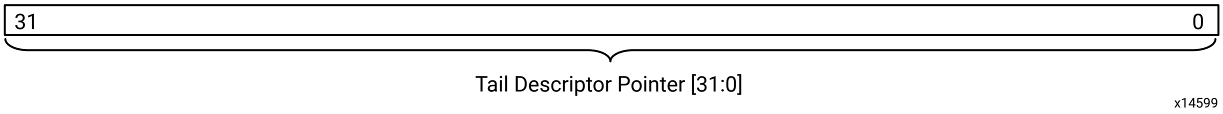


Table 21: **S2MM_TAILDESC_MSB Register Details**

Bits	Field Name	Default Value	Access Type	Description
31 to 0	Tail Descriptor Pointer	zeros	R/W	Indicates the pause pointer in a descriptor chain. The AXI DMA SG Engine pauses descriptor fetching after completing operations on the descriptor whose current descriptor pointer matches the tail descriptor pointer. When AXI DMA Channel is not halted (DMASR.Halted = 0), a write by the CPU to the TAILDESC_PTR_MSB register causes the AXI DMA SG Engine to start fetching descriptors or restart if it was idle (DMASR.Idle = 1). If it was not idle, then writing to TAILDESC_PTR has no effect except to reposition the pause point. If the AXI DMA Channel DMACR.RS bit is set to 0 (DMASR.Halted =1 and DMACR.RS = 0), a write by the CPU to the TAILDESC_PTR register has no effect except to reposition the pause point. ***Note*:** The software must not move the Tail Pointer to a location that has not been updated. The software processes and reallocates all completed descriptors (Cmplted = 1), clears the completed bits and then moves the tail pointer. The software must move the pointer to the last descriptor it updated.

S2MM_DA (S2MM DMA Destination Address Register – Offset 48h)

This register provides the Destination Address for writing to system memory for the Stream to Memory Map to DMA transfer.

Figure 19: **S2MM_DA Register**

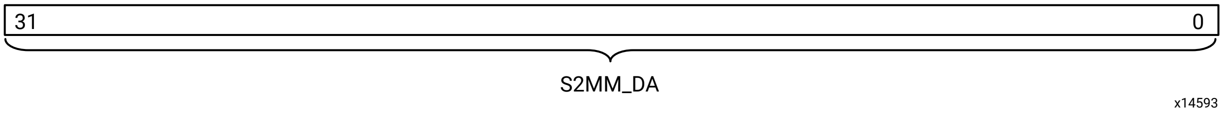


Table 22: **S2MM_DA Register Details**

Bits	Field Name	Default Value	Access Type	Description
31 to 0	Destination Address	zeros	R/W	Indicates the destination address the AXI DMA writes to transfer data from AXI4-Stream on S2MM Channel. ***Note*:** If Data Realignment Engine is included, the Destination Address can be at any byte offset. If Data Realignment Engine is not included, the Destination Address must be S2MM Memory Map data width aligned.

S2MM_DA_MSB (S2MM DMA Destination Address Register – Offset 4Ch)

This register provides the upper 32 bits of Destination Address for writing to system memory for the Stream to Memory Map to DMA transfer. This is used only when DMA is configured for address space greater than 32.

Figure 20: **S2MM_DA_MSB Register**

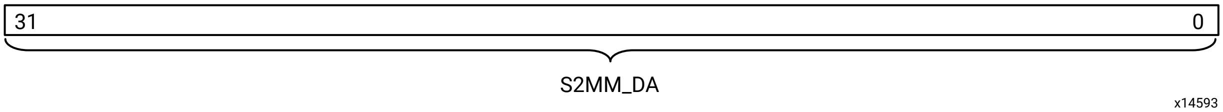


Table 23: **S2MM_DA_MSB Register Details**

Bits	Field Name	Default Value	Access Type	Description
31 to 0	Destination Address	zeros	R/W	Indicates the MSB 32 bits of the destination address the AXI DMA writes to transfer data from AXI4-Stream on the S2MM Channel. ***Note*:** If Data Realignment Engine is included, the Destination Address can be at any byte offset. If Data Realignment Engine is not included, the Destination Address must be S2MM Memory Map data width aligned.

S2MM_LENGTH (S2MM DMA Buffer Length Register – Offset 58h)

This register provides the length in bytes of the buffer to write data from the Stream to Memory map DMA transfer.

Figure 21: **S2MM_LENGTH Register**

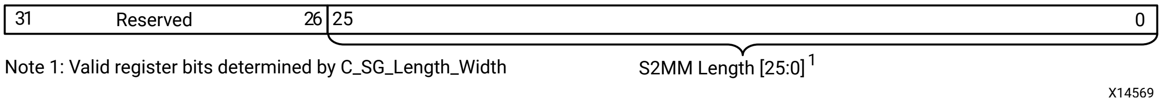


Table 24: **S2MM_LENGTH Register Details**

Bits	Field Name	Default Value	Access Type	Description
25[1] to 0	Length	zeros	R/W	Indicates the length in bytes of the S2MM buffer available to write receive data from the S2MM channel. Writing a non-zero value to this register enables S2MM channel to receive packet data. At the completion of the S2MM transfer, the number of actual bytes written on the S2MM AXI4 interface is updated to the S2MM_LENGTH register. ***Note*:** This value must be greater than or equal to the largest expected packet to be received on S2MM AXI4-Stream. Values smaller than the received packet result in undefined behavior.
31 to 26	Reserved	0	RO	Writing to these bits has no effect and they are always read as zeros.

Notes:

1. Width of Length field determined by Buffer Length Register Width parameter. Minimum width is 8 bits (7 to 0) and maximum width is 26 bits (25 to 0).

Scatter Gather Descriptor

This section defines the fields of the S2MM (Receive) and MM2S (Transmit) Scatter Gather Descriptors for when the AXI DMA is configured for Scatter/Gather Mode. The descriptor is made up of eight 32-bit base words and 0 or 5 User Application words. The descriptor has future support for 64-bit addresses and support for user application data. Multiple descriptors per packet are supported through the Start of Frame and End of Frame flags. Completed status and Interrupt on Complete are also included. The Buffer Length can describe up to 67,108,863 bytes of data buffer per descriptor. Two descriptor chains are required for the two data transfer directions, MM2S and S2MM.

Table 25: **Descriptor Fields (Non-multichannel Mode)**

Address Space Offset[1]	Name	Description
00h	NXTDESC	Next Descriptor Pointer
04h	NXTDESC_MSB	Upper 32 bits of Next Descriptor Pointer
08h	BUFFER_ADDRESS	Buffer Address
0Ch	BUFFER_ADDRESS_MSB	Upper32 bits of Buffer Address.
10h	RESERVED	N/A
14h	RESERVED	N/A
18h	CONTROL	Control
1Ch	STATUS	Status
20h	APP0	User Application Field 0[2]
24h	APP1	User Application Field 1
28h	APP2	User Application Field 2
2Ch	APP3	User Application Field 3
30h	APP4	User Application Field 4

Notes:

1. Address Space Offset is relative to 16 - 32-bit word alignment in system memory, that is, 0x00, 0x40, 0x80, and so forth.
2. User Application fields (APP0, APP1, APP2, APP3, and APP4) are only used when the Control/Status Streams are included, When the Control/Status Streams are not included, the User Application fields are not fetched or updated by the Scatter Gather Engine.
3. The MSB fields or the upper 32-bit addresses are used only when DMA is configured for an address space greater than 32.

MM2S_NXTDESC (MM2S Next Descriptor Pointer)

This value provides the pointer to the next descriptor in the descriptor chain.

Figure 22: **MM2S_NXTDESC**

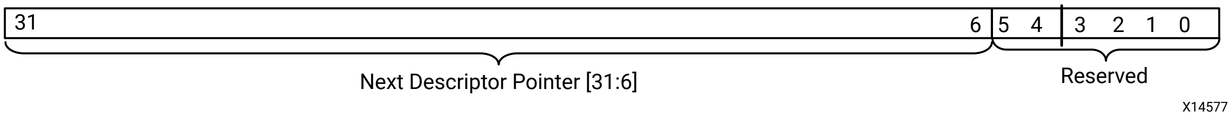


Table 26: **MM2S_NXTDESC Details**

Bits	Field Name	Description
5 to 0	Reserved	These bits are reserved and should be set to zero.
31 to 6	Next Descriptor Pointer	Indicates the lower order pointer pointing to the first word of the next descriptor. ***Note*:** Descriptors must be 16-word aligned, that is, 0x00, 0x40, 0x80, and so forth. Any other alignment has undefined results.

MM2S_NXTDESC_MSB (MM2S Next Descriptor Pointer)

This value provides the upper 32 bits of the pointer to the next descriptor in the descriptor chain. This is used only when AXI DMA is configured for an address space greater than 32.

Figure 23: **MM2S_NXTDESC_MSB**

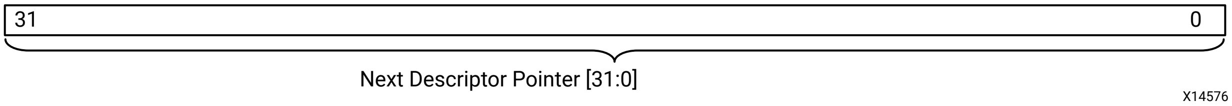


Table 27: **MM2S_NXTDESC_MSB Details**

Bits	Field Name	Description
31 to 0	Next Descriptor Pointer	Indicates the MSB 32 bits of the pointer pointing to the first word of the next descriptor.

MM2S_BUFFER_ADDRESS (MM2S Buffer Address)

This value provides the pointer to the buffer of data to transfer from system memory to stream.

Figure 24: **MM2S Buffer Address**

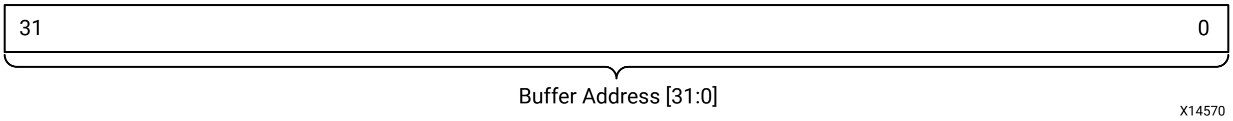


Table 28: **MM2S_BUFFER_ADDRESS Details**

Bits	Field Name	Description
31 to 0	Buffer Address	Provides the location of the data to transfer from Memory Map to Stream. ***Note***: If Data Realignment Engine is included, the Buffer Address can be at any byte offset, but data within a buffer must be contiguous. If the Data Realignment Engine is not included, the Buffer Address must be MM2S Memory Map data-width aligned.

MM2S_BUFFER_ADDRESS_MSB (MM2S Buffer Address)

This value provides the upper 32 bits of pointer to the buffer of data to transfer from system memory to stream. This is used only when AXI DMA is configured for an address space greater than 32.

Figure 25: **MM2S Buffer Address**

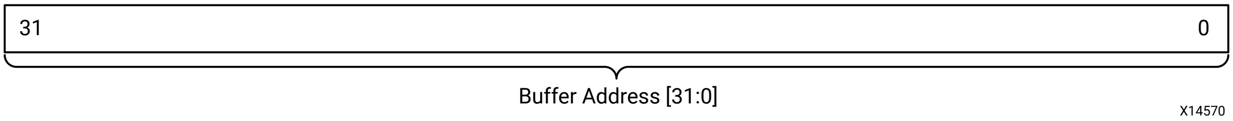


Table 29: **MM2S_BUFFER_ADDRESS_MSB Details**

Bits	Field Name	Description
31 to 0	Buffer Address	Provides the MSB 32 bits of the location of the data to transfer from Memory Map to Stream.

MM2S_CONTROL (MM2S Control)

This value provides control for MM2S transfers from memory map to stream.

Figure 26: **MM2S_CONTROL**

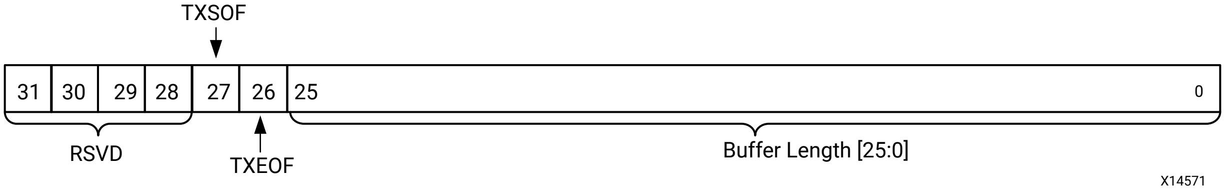


Table 30: **MM2S_CONTROL Details**

Bits	Field Name	Description
25 to 0	Buffer Length	Indicates the size in bytes of the transfer buffer. This value indicates the amount of bytes to transmit out on the MM2S stream. The usable width of buffer length is specified by the parameter, Width of Buffer Length Register. A maximum of 67,108,863 bytes of transfer can be described by this field. When configuring the AXI_DMA in Micro mode, this value should not exceed the following equation: (MM2S Memory Mapped Data width/8)*Burst_length - 1 ***Note***: Setting the buffer length register width smaller than 26 reduces FPGA resource utilization.
26	Transmit End Of Frame (TXEOF)	End of Frame. Flag indicating the last buffer to be processed. This flag is set by the CPU to indicate to AXI DMA that this descriptor describes the end of the packet. The buffer associated with this descriptor is transmitted last. • 0= Not End of Frame. • 1= End of Frame. ***Note***: For proper operation, there must be a Start of Frame (SOF) descriptor (TXSOF=1) and an End of Frame (EOF) descriptor (TXEOF=1) per packet. It is valid to have a single descriptor describe an entire packet that is a descriptor with both TXSOF=1 and TXEOF=1.
27	TXSOF	Start of Frame. Flag indicating the first buffer to be processed. This flag is set by the CPU to indicate to AXI DMA that this descriptor describes the start of the packet. The buffer associated with this descriptor is transmitted first. • 0= Not start of frame. • 1= Start of frame. ***Note***: When Status Control Stream is enabled, user application data from APP0 to APP4 of the Start of Frame (SOF) descriptor (TXSOF=1) is transmitted on the control stream output.
31 to 28	Reserved	This bit is reserved and should be written as zero.

MM2S_STATUS (MM2S Status)

This value provides status for MM2S transfers from memory map to stream.

Figure 27: **MM2S_STATUS**

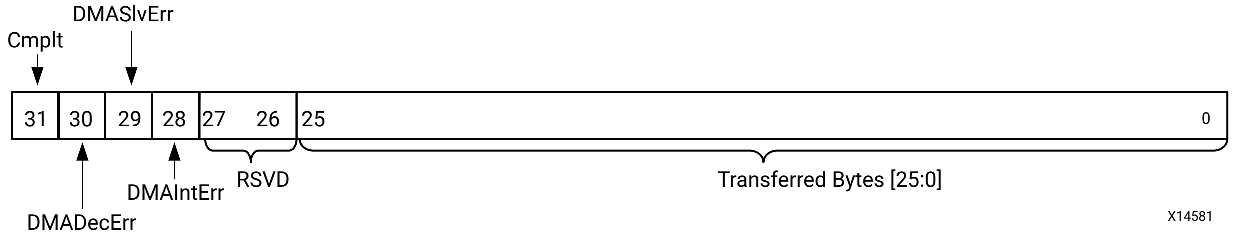


Table 31: **MM2S_STATUS Details**

Bits	Field Name	Description
25 to 0	Transferred Bytes	Indicates the size in bytes of the actual data transferred for this descriptor. This value indicates the amount of bytes to transmit out on MM2S stream. This value should match the Control Buffer Length field. The usable width of Transferred Bytes is specified by the parameter, Width of Buffer Length Register. A maximum of 67,108,863 bytes of transfer can be described by this field. AXI_DMA does not update these fields when configured in Micro mode. ***Note***: Setting the Buffer length Register Width smaller than 26 reduces FPGA resource utilization. This field is not updated when AXI_DMA is configured in Micro mode.
27 to 26	Reserved	These bits are reserved and should be set to zero.
28	DMAIntErr	DMA Internal Error. Internal Error detected by primary AXI Data Mover. This error can occur if a 0 length bytes to transfer is fed to the AXI Data Mover. This only happens if the buffer length specified in the fetched descriptor is set to 0. This error condition causes the AXI DMA to halt gracefully. The DMACR.RS bit is set to 0, and when the engine has completely shut down, the DMASR.Halted bit is set to 1. • 0 = No DMA Internal Errors. • 1 = DMA Internal Error detected. DMA Engine halts.
29	DMASlvErr	DMA Slave Error. Slave Error detected by primary AXI Data Mover. This error occurs if the slave read from the Memory Map interface issues a Slave Error. This error condition causes the AXI DMA to halt gracefully. The DMACR.RS bit is set to 0, and when the engine has completely shut down, the DMASR.Halted bit is set to 1. • 0 = No DMA Slave Errors. • 1 = DMA Slave Error detected. DMA Engine halts.
30	DMADecErr	DMA Decode Error. Decode Error detected by primary AXI Data Mover. This error occurs if the Descriptor Buffer Address points to an invalid address. This error condition causes the AXI DMA to halt gracefully. The DMACR.RS bit is set to 0, and when the engine has completely shut down, the DMASR.Halted bit is set to 1. • 0 = No DMA Decode Errors. • 1 = DMA Decode Error detected. DMA Engine halts.
31	Cmplt	Completed. This indicates to the software that the DMA Engine has completed the transfer as described by the associated descriptor. The DMA Engine sets this bit to 1 when the transfer is completed. The software might manipulate any descriptor with the Completed bit set to 1. • 0 = Descriptor not completed. • 1 = Descriptor completed. ***Note***: If a descriptor is fetched with this bit set to 1, the descriptor is considered a stale descriptor. An SGIntErr is flagged, and the AXI DMA engine halts.

MM2S_APP0 to MM2S_APP4

This value provides User Application fields for MM2S control stream.

Figure 28: **MM2S_STATUS**

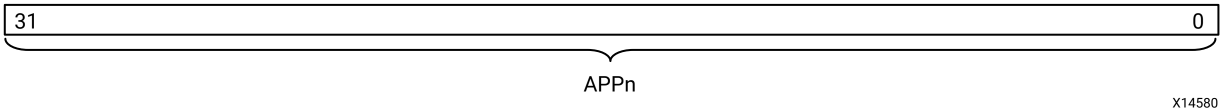


Table 32: **User Application Details**

Bits	Field Name	Description
31 to 0	APP0 to APP4	User application fields 0 to 4. Specifies user-specific application data. When Status Control Stream is enabled, the Application (APP) fields of the Start of Frame (SOF) Descriptor are transmitted to the AXI Control Stream. For other MM2Sdescriptors with SOF = 0, the APP fields are fetched but ignored. ***Note*:** These fields are not fetched when the Status Control Stream is not enabled.

S2MM_NXTDESC (S2MM Next Descriptor Pointer)

This value provides the pointer to the next descriptor in the descriptor chain.

Figure 29: **S2MM_NXTDESC**

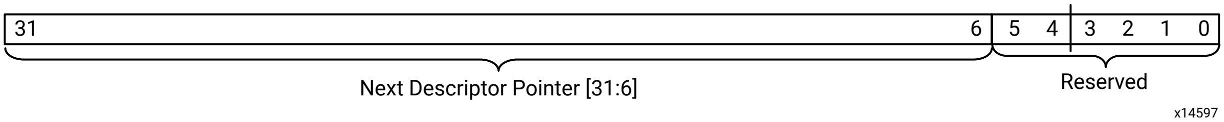


Table 33: **S2MM_NXTDESC Details**

Bits	Field Name	Description
5 to 0	Reserved	These bits are reserved and should be set to zero.
31 to 6	Next Descriptor Pointer	Indicates the lower order pointer pointing to the first word of the next descriptor. ***Note*:** Descriptors must be 16-word aligned, that is, 0x00, 0x40, 0x80, and so forth. Any other alignment has undefined results.

S2MM_NXTDESC_MSB (S2MM Next Descriptor Pointer)

This value provides the upper 32 bits of the pointer to the next descriptor in the descriptor chain. This is used only when AXI DMA is configured for an address space greater than 32.

Figure 30: **S2MM_NXTDESC_MSB**

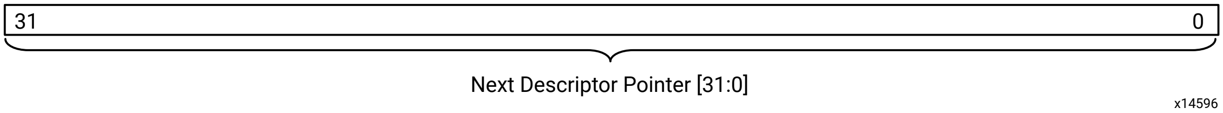


Table 34: **S2MM_NXTDESC_MSB Details**

Bits	Field Name	Description
31 to 0	Next Descriptor Pointer	Indicates the MSB 32 bits of the pointer pointing to the first word of the next descriptor.

S2MM_BUFFER_ADDRESS (S2MM Buffer Address)

This value provides the pointer to the buffer space available to transfer data from stream to system memory.

Figure 31: **S2MM Buffer Address**

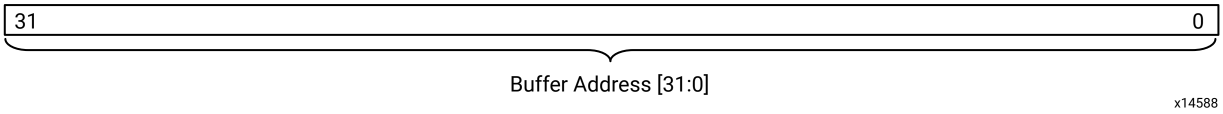


Table 35: **S2MM_BUFFER_ADDRESS Details**

Bits	Field Name	Description
31 to 0	Buffer Address	Provides the location of the buffer space available to store data transferred from Stream to Memory Map. ***Note*:** If Data Realignment Engine is included, the Buffer Address can be at any byte offset. If Data Realignment Engine is not included the Buffer Address must be S2MM Memory Mapped data width aligned.

S2MM_BUFFER_ADDRESS_MSB (S2MM Buffer Address)

This value provides the upper 32 bits of the pointer to the buffer space available to transfer data from stream to system memory. This is used only when AXI DMA is configured for an address space greater than 32.

Figure 32: **S2MM Buffer Address (MSB)**

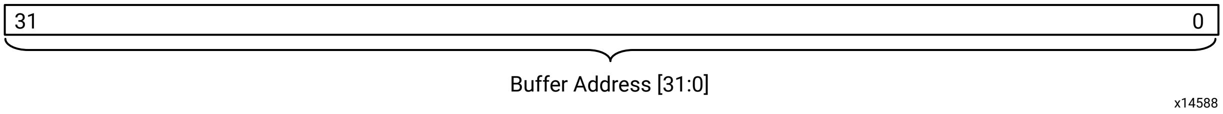


Table 36: **S2MM_BUFFER_ADDRESS_MSB Details**

Bits	Field Name	Description
31 to 0	Buffer Address	Provides the MSB 32 bits of the location of the buffer space available to store data transferred from Stream to Memory Map.

S2MM_CONTROL (S2MM Control)

This value provides control for S2MM transfers from stream to memory map.

Figure 33: **S2MM_CONTROL**

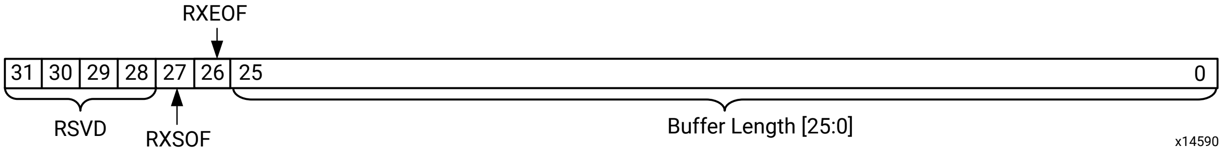


Table 37: **S2MM_CONTROL Details**

Bits	Field Name	Description
31 to 28	Reserved	These bits are reserved and should be set to zero.
27	RXSOF	Start of Frame. Flag indicating the first buffer to be processed. This flag is set by the sw/user to indicate to AXI DMA that this descriptor describes the start of the packet. The buffer associated with this descriptor is received first. • 0 = Not Start of Frame. • 1 = Start of Frame. This is applicable only when AXI_DMA is configured in Micro mode.
26	Receive End Of Frame	End of Frame. Flag indicating the last buffer to be processed. This flag is set by the sw/user to indicate to AXI DMA that this descriptor describes the end of the packet. The buffer associated with this descriptor is received last. • 0 = Not End of Frame. • 1 = End of Frame. This is applicable only when AXI_DMA is configured in Micro mode.
25 to 0	Buffer Length	This value indicates the amount of space in bytes available for receiving data in an S2MM stream. The usable width of buffer length is specified by the parameter, Width of Buffer Length Register. A maximum of 67,108,863 bytes of transfer can be described by this field. ***Note***: The total buffer space in the S2MM descriptor chain (that is, the sum of buffer length values for each descriptor in a chain) must be, at a minimum, capable of holding the maximum receive packet size. Undefined results occur if a packet larger than the defined buffer space is received. ***Note***: Setting the Buffer Length Register Width smaller than 26 reduces FPGA resource utilization. ***Note***: When configuring the AXI_DMA in Micro mode, this value should not exceed the following equation: (S2MM Memory Mapped Datawidth/8)*Burst_length - 1

S2MM_STATUS (S2MM Status)

This value provides status for S2MM transfers from stream to memory map.

Figure 34: **S2MM_STATUS**

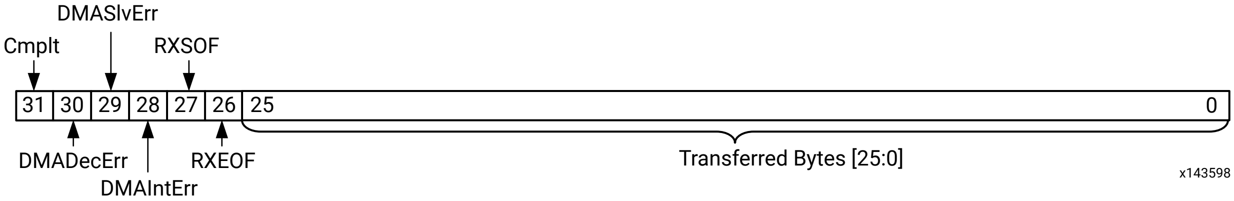


Table 38: **S2MM_STATUS Description**

Bits	Field Name	Description
25 to 0	Transferred Bytes	This value indicates the amount of data received and stored in the buffer described by this descriptor. This might or might not match the buffer length. For example, if this descriptor indicates a buffer length of 1,024 bytes but only 50 bytes were received and stored in the buffer, then the Transferred Bytes field indicates 0x32. The entire receive packet length can be determined by adding the Transferred Byte values from each descriptor from the RXSOF descriptor to the Receive End of Frame (RXEOF) descriptor. ***Note***: The usable width of Transferred Bytes is specified by the parameter, Width of Buffer Length Register. A maximum of 67,108,863 bytes of transfer can be described by this field. ***Note***: Setting the Buffer Length Register Width smaller than 26 reduces FPGA resource utilization. This field is not updated when AXI_DMA is configured in Micro mode.
26	RXEOF	End of Frame. Flag indicating buffer holds the last part of packet. This bit is set by AXI DMA to indicate to the sw/user that the buffer associated with this descriptor contains the end of the packet. • 0= Not End of Frame. • 1= End of Frame. ***Note***: User Application data sent through the status stream input is stored in APP0 to APP4 of the RXEOF descriptor when the Control/Status Stream is enabled.
27	RXSOF	Start of Frame. Flag indicating buffer holds first part of packet. This bit is set by AXI DMA to indicate to the sw/user that the buffer associated with this descriptor contains the start of the packet. • 0= Not start of frame. • 1= Start of frame.
28	DMAIntErr	DMA Internal Error. Internal Error detected by primary AXI Data Mover. This error can occur if a 0 length bytes to transfer is fed to the AXI Data Mover. This only happens if the Buffer Length specified in the fetched descriptor is set to 0. This error can also be caused if an under-run or over-run condition. This error condition causes the AXI DMA to halt gracefully. The DMACR.RS bit is set to 0, and when the engine has completely shut down, the DMASR.Halted bit is set to 1. • 0= No DMA Internal Errors. • 1= DMA Internal Error detected. DMA Engine halts.
29	DMASlvErr	DMA Slave Error. Slave Error detected by primary AXI Data Mover. This error occurs if the slave read from the Memory Map interface issues a Slave Error. This error condition causes the AXI DMA to halt gracefully. The DMACR.RS bit is set to 0, and when the engine has completely shut down, the DMASR.Halted bit is set to 1. • 0= No DMA Slave Errors. • 1= DMA Slave Error detected. DMA Engine halts.

Table 38: **S2MM_STATUS Description** *(cont'd)*

Bits	Field Name	Description
30	DMADecErr	DMA Decode Error. Decode Error detected by primary AXI Data Mover. This error occurs if the Descriptor Buffer Address points to an invalid address. This error condition causes the AXI DMA to halt gracefully. The DMACR.RS bit is set to 0, and when the engine has completely shut down, the DMASR.Halted bit is set to 1. • 0= No DMA Decode Errors. • 1= DMA Decode Error detected. DMA Engine halts.
31	Cmplt	Completed. This indicates to the software that the DMA Engine has completed the transfer as described by the associated descriptor. The DMA Engine sets this bit to 1 when the transfer is completed. The software can manipulate any descriptor with the Completed bit set to 1. • 0= Descriptor not completed. • 1= Descriptor completed. ***Note***: If a descriptor is fetched with this bit set to 1, the descriptor is considered a stale descriptor. An SGIntErr is flagged and the AXI DMA engine halts.

S2MM_APP0 to S2MM_APP3 (S2MM User Application Fields 0 to 3)

This value provides User Application field space for the S2MM received status on the Status Stream.

Figure 35: **S2MM_APP0 to S2MM_APP3**

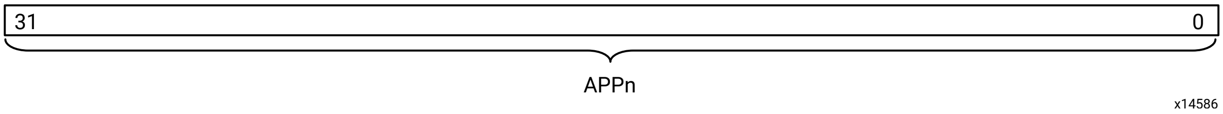


Table 39: **User Application 0 to 3 Details**

Bits	Field Name	Description
31 to 0	APP0 to APP3	When Status/Control Stream is enabled, the status data received on the AXI Status Stream is stored into the APP fields of the End of Frame (EOF) Descriptor. For other S2MM descriptors with EOF = 0, the APP fields are set to zero by the Scatter Gather Engine. ***Note***: These fields are not updated by the Scatter Gather Engine if the Status/Control Fields are disabled.

S2MM_APP4 (S2MM User Application Field 4)

This value provides User Application 4 field space for S2MM received status on the Status Stream.

Figure 36: **S2MM_APP4**

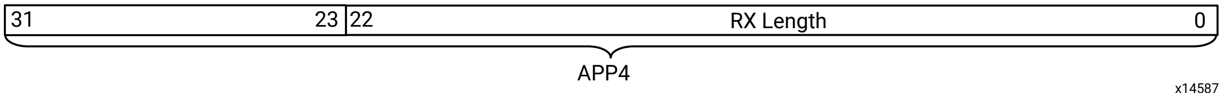


Table 40: **User Application 4 Details**

Bits	Field Name	Description
31 to 0	APP4/RxLength	User Application field 4 and Receive Byte Length. If Use RxLength In Status Stream is not enabled, this field functions identically to APP0 to APP3 in that the status data received on the AXI Status Stream is stored into the APP4 field of the End of Frame (EOF) Descriptor. This field has a dual purpose when Use RxLength in Status Stream is enabled. The first least significant bits specified in the Buffer Length Register Width specify the total number of receive bytes for a packet that were received on the S2MM primary data stream. Second, the remaining most significant bits are User Application data.

Descriptor Management

Prior to starting DMA operations, the software application must set up a descriptor chain. When the AXI DMA begins processing the descriptors, it fetches, processes, and then updates the descriptors. By analyzing the descriptors, the software application can read the status on the associated DMA transfer, fetch user information on receive (S2MM) channels, and determine completion of the transfer. With this information, the software application can manage the descriptors and data buffers.

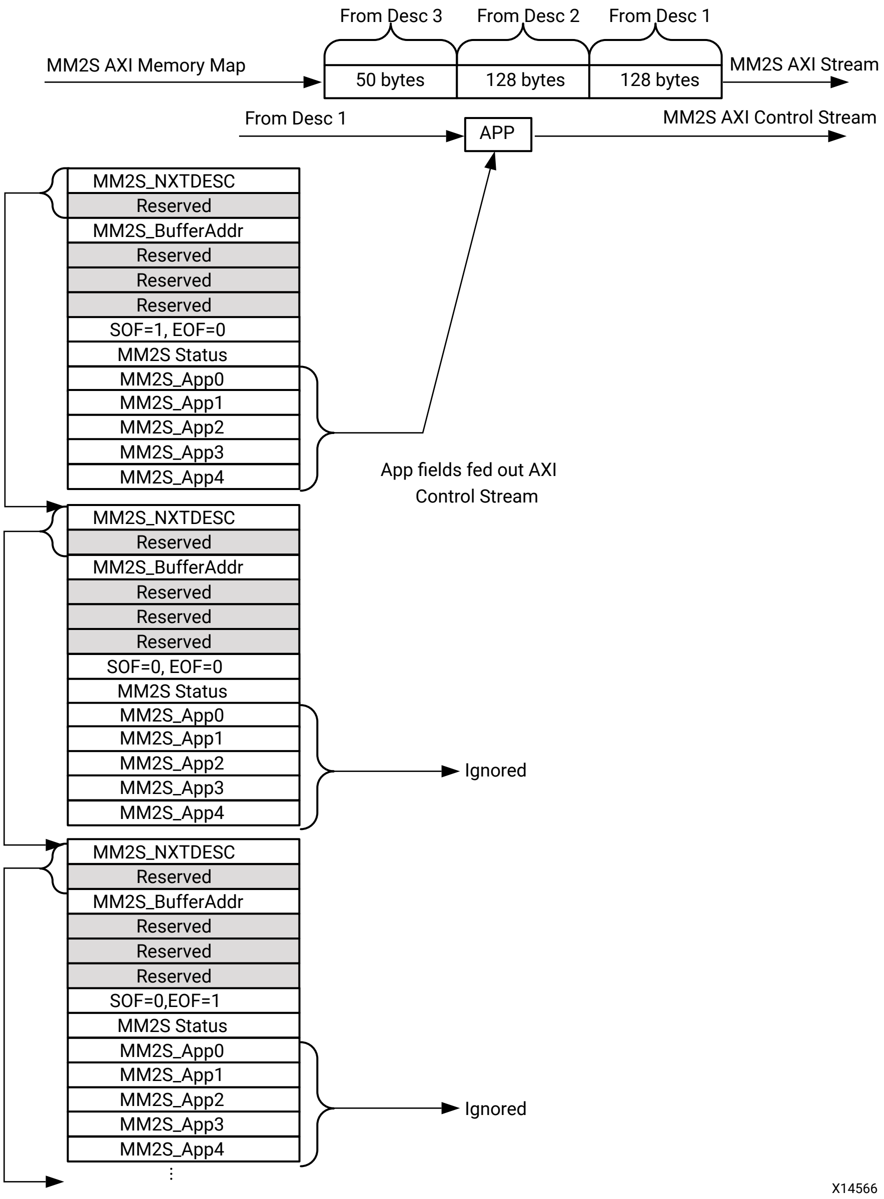
Software applications process each buffer associated with completed descriptors and reallocate the descriptor for AXI DMA use. To prevent software and hardware from stepping on each other, a Tail Pointer Mode is created. The tail pointer is initialized by software to point to the end of the descriptor chain. This becomes the pause point for hardware. When hardware begins running, it fetches and processes each descriptor in the chain until it reaches the tail pointer. The AXI DMA then pauses descriptor processing. The software is allowed to process and re-allocate any descriptor whose Complete bit is set to 1.

The act of writing to the TAILDESC register causes the AXI DMA hardware, if it is paused at the tail pointer, to begin processing descriptors again. If the AXI DMA hardware is not paused at the TAILDESC pointer, writing to the TAILDESC register has no effect on the hardware. In this situation, the AXI DMA continues to process descriptors until reaching the new tail descriptor pointer location. Descriptor Management must be done by the software. AXI DMA does not manage the descriptors.

MM2S Descriptor Settings and AXI Control Stream

The relationship between descriptor SOF/EOF settings and the AXI Control Stream is illustrated in the following figure. The descriptor with SOF=1 is the beginning of the packet and resets DRE for the MM2S direction. The User Application fields for this descriptor are also presented on the AXI Control Stream if the Status/Control Stream is enabled. User Application fields following a descriptor with SOF=1, up to and including the descriptor with EOF =1, are ignored by the AXI DMA engine. If Status/Control Stream is disabled, the User Application fields are not fetched by the SG Fetch Engine.

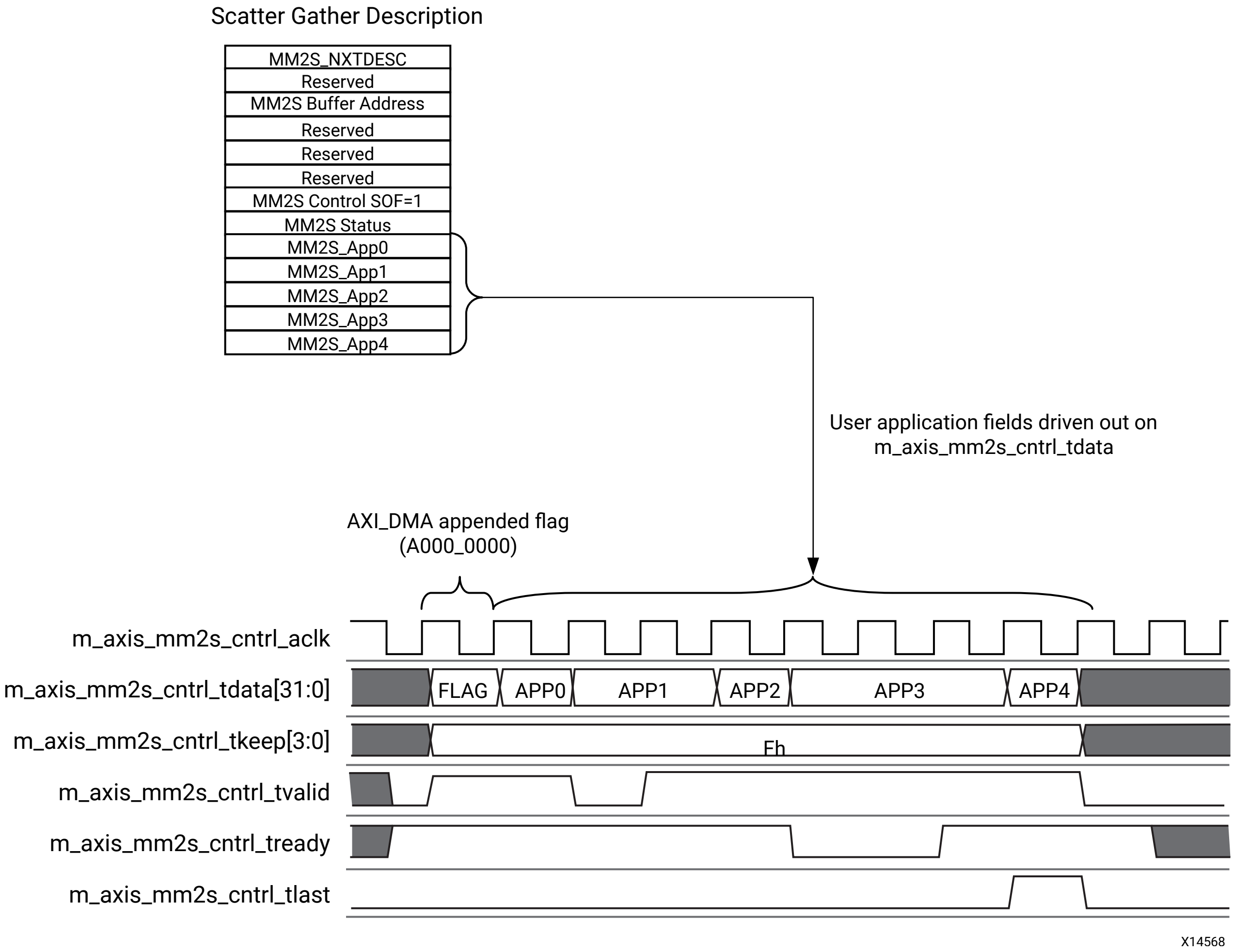
Figure 37: **Detail of Descriptor Relationship to MM2S Stream and Control Stream**



AXI Control Stream

The AXI control stream is provided from the Scatter Gather Descriptor to a target device for User Application data. The control data is associated with the MM2S primary data stream and can be sent out of AXI DMA prior to, during, or after the primary data packet. Throttling by the target device is allowed, and throttling by AXI DMA can occur. The following figure shows an example of how descriptor User Application fields are presented on the AXI control stream. AXI DMA inserts a flag indicating the data type to the target device. This is sent as the first word. For Ethernet, the control tag is 0xA in the four Most Significant Bits (MSBs) of the first word.

Figure 38: **Example User Application Field/Timing for MM2S Control Stream**

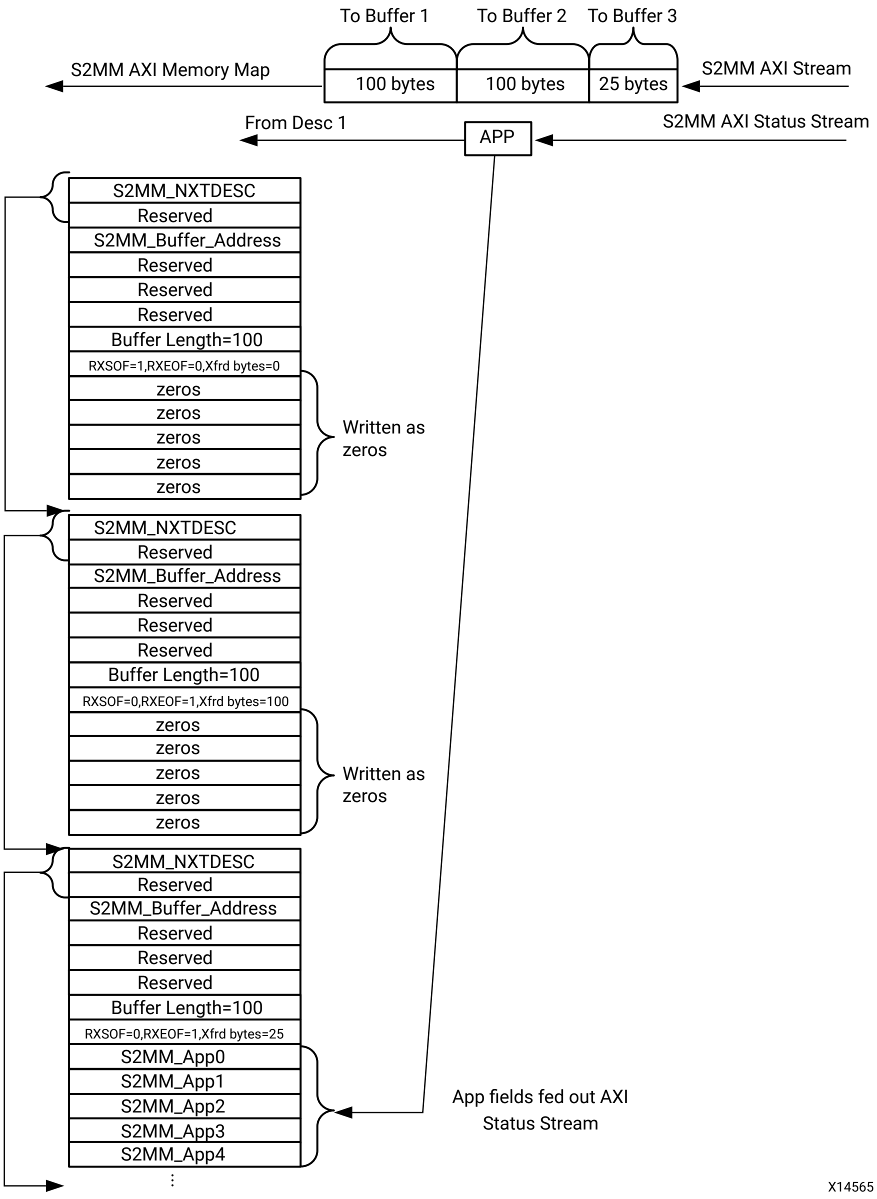


S2MM Descriptor Settings and AXI Status Stream

The relationship between descriptor RXSOF/RXEOF settings and the AXI Status Stream are illustrated in the following figure. The descriptor with RXSOF=1 describes the buffer containing the first part of the receive packet. The Descriptor with RXEOF=1 describes the buffer containing the last part of the receive packet.

For proper operation, the software must specify enough buffer space (the sum of the buffer lengths in each descriptor of the descriptor chain) to be greater than or equal to the maximum sized packet that is received.

Figure 39: **Detail of Descriptor Relationship to S2MM Stream and Status Stream**



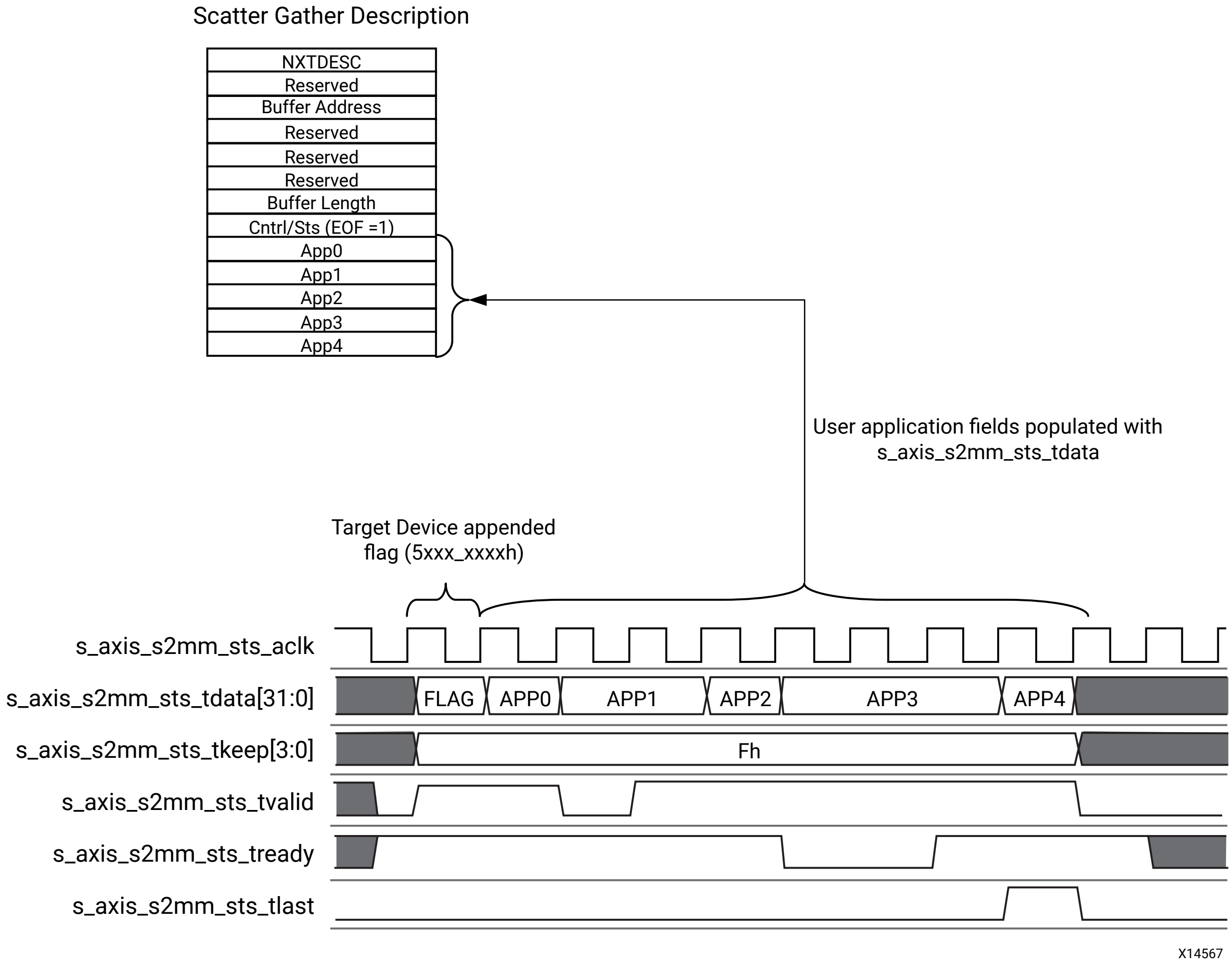
If the Status/Control Stream is included, the status received is stored in the User Application fields (APP0 to APP4) of the descriptor with RXEOF set.

The actual byte count of received and stored data for a particular buffer is updated to the Transferred Bytes field in the associated descriptor. The software can determine how many bytes were received by walking the descriptors from RXSOF to RXEOF and adding the Bytes Transferred fields to get a total byte count. For applications where you provide the total length in the status stream, this value is stored in the user-defined application location in the descriptor with RXEOF=1.

AXI Status Stream

The AXI status stream is provided for transfer of target device status to User Application data fields in the Scatter Gather descriptor. The status data is associated with the S2MM primary data stream. As shown in the following figure, the status packet updates to the app fields of the detected last descriptor (RXEOF = 1) describing the packet. Normally, the status stream should come at the start of the S2MM data stream. If the Use RxLength In Status Stream is disabled, then the status stream can come at any time during the course of the S2MM frame. The End of Frame (EOF) Buffer Descriptor (BD) update would happen only when the entire status stream is received.

Figure 40: **Example User Application Field/Timing for S2MM Status Stream**



Multichannel DMA Support

> **IMPORTANT!** *The Multichannel support from AXI DMA will be discontinued. For MultiChannel support, see the AXI Multichannel Direct Memory Access LogiCORE IP Product Guide (PG288).*

Multichannel mode enables DMA to connect to multiple masters and slaves on the streaming side. A new set of signals associated with source and destination signaling are added. They are:

- tid – 5-bit signal. User-defined sideband signaling.
- tdest – 5-bit signal. Provides coarse routing information for the data stream.
- tuser – 4-bit signal. User defined sideband signaling.

Scatter Gather Mode (C_INCLUDE_SG = 1)

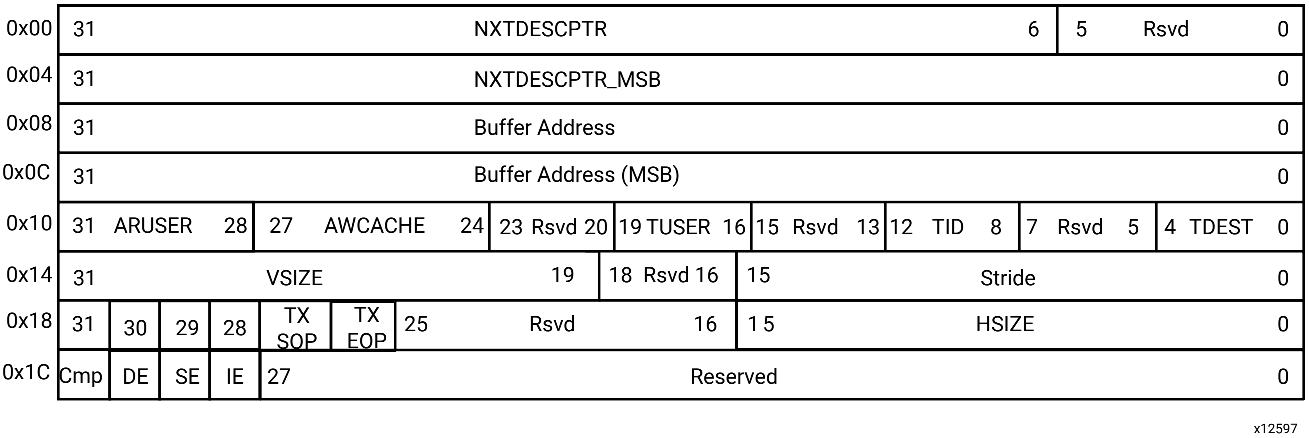
New descriptor fields are added to support multichannel and 2-D transfers. As described in AXI DMA Multichannel Operation, AXI DMA supports efficient two-dimensional memory access patterns, transferring 2-D blocks across the AXI4-Stream channel. Memory access patterns are controlled with three parameters: HSIZE, VSIZE, and STRIDE. Multiple descriptors per packet are supported through the Start of Packet and End of Packet flags.

In this mode, the AMD Vivado™ Integrated Design Environment (IDE) IP customization feature disables the Status/Control Stream.

AXI DMA can be set in multichannel mode by enabling the Multi Channel Mode and selecting the required number of channels on MM2S and S2MM paths.

MM2S (TX) Descriptor

Figure 41: **TX Descriptor**



0x00	31 NXTDESCPTR 6	5 Rsvd 0						
0x04	31 NXTDESCPTR_MSB 0							
0x08	31 Buffer Address 0							
0x0C	31 Buffer Address (MSB) 0							
0x10	31 ARUSER 28	27 AWCACHE 24	23 Rsvd 20	19 TUSER 16	15 Rsvd 13	12 TID 8	7 Rsvd 5	4 TDEST 0
0x14	31 VSIZE 19	18 Rsvd 16	15 Stride 0					
0x18	31	30	29	28	TX SOP	TX EOP	25 Rsvd 16	15 HSIZE 0
0x1C	Cmp	DE	SE	IE	27 Reserved 0			

TX Descriptor Fields

Table 41: **TX Descriptor Fields**

Address Offset	Name	Description
00h	NXTDESC	Bits 5:0 – Reserved Bits 31:6 – Next Descriptor Pointer
04h	NXTDESC MSB	Provides the upper 32 bits of the next descriptor pointer. Applicable when AXI DMA is configured for an address space greater than 32.
08h	BUFFER_ADDRESS	Bits 31:0 – Buffer Address Provides the location of the data to transfer from Memory Map to Stream. The address should be aligned to the Memory Map data width.
0Ch	BUFFER_ADDRESS	Provides the upper 32 bits of buffer address. This is applicable only when AXI DMA is configured for an address space greater than 32.
		Multichannel Control bits. Bits 4:0 – TDEST provides routing information for the data stream. TDEST values are static for the entire packet. TDEST values provided in the TX descriptor field are presented on TDEST signals of streaming side.
		Bits 7:5 – Reserved
		Bits 12:8 – TID: Provides a stream identifier. TID values are static for entire packet. TID values provided in the TX descriptor field are presented on TID signals of the streaming side.
		Bits 15:13 – Reserved
		Bits 19:16 – TUSER: Sideband signals used for user-defined information. TUSER values are static for entire packet. TUSER values provided in the TX descriptor field are presented on TUSER signals of streaming side.
10h	MC_CTL	Bits 23:20 – Reserved
		Bits 27:24 – ARCACHE: Cache type. This signal provides additional information about the cacheable characteristics of the transfer. See the *AMBA® AXI and ACE Protocol Specification* for a different decoding mechanism. ARCACHE values from TX descriptor are presented on ARCACHE [3:0] bus during address cycle. Default value of this field is 0011.
		Bits 31:28 – ARUSER: Sideband signals used for user-defined information. ARUSER values from TX descriptor are presented on ARUSER [3:0]. ARUSER values and their interpretations are user-defined. You can keep ARUSER static for the entire packet by programming the same values in all the descriptors within a chain.
14h	STRIDE_VSIZE	Bits 15:0 – Stride Control. It is the address distance between the first address of successive “horizontal” reads. Reads will start at the Buffer Address and read HSIZE bytes, then skip STRIDE-HSIZE addresses and read HSIZE bytes, and so on. This continues until VSIZE lines have been read. On AXI4-Stream this is transmitted out on the m_axis_mm2s_ interface as one contiguous packet and is terminated with a single assertion of TLAST on the last data beat of the transfer. Bits 18:16 – Reserved Bits 31:19 – Number of “horizontal lines” for stride access. Can represent two-dimensional video data or the size of a 2-D matrix. This is the number of transfers, each HSIZE bytes long, that are expected to be transmitted for each packet.

Table 41: **TX Descriptor Fields** *(cont'd)*

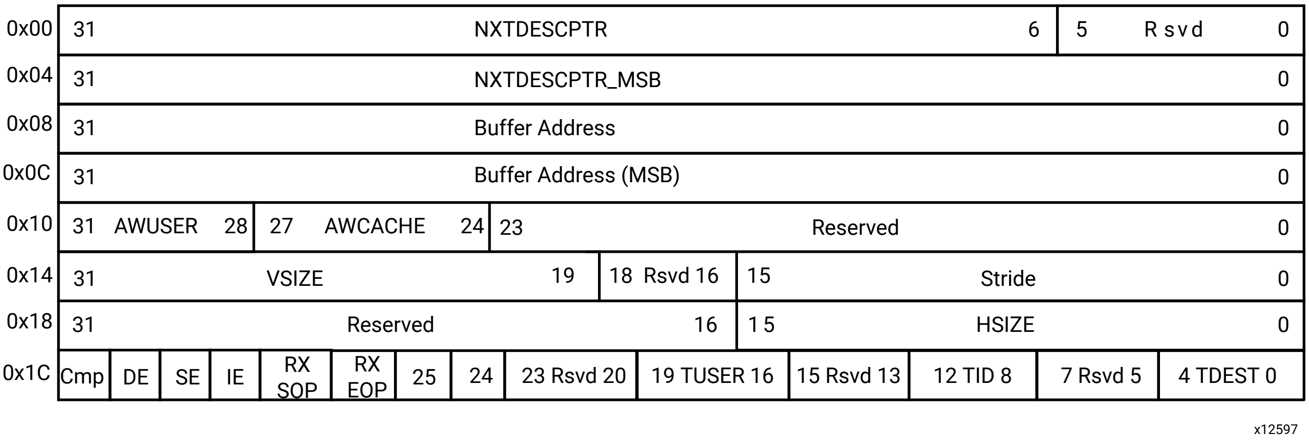
Address Offset	Name	Description
18h	HSIZE	Bits 15:0 – Number of bytes to transfer in each “horizontal line” from successive contiguous byte addresses. Can represent a portion of a video line or a portion of a matrix row when the matrix is read in row major order. Bits 25:16 – Reserved Bit 26 – TXEOP – End of packet flag. It indicates the buffer associated with this descriptor is transmitted last. This flag is set by the CPU. 0 – Not end of packet. 1 – End of packet Bit 27 – TXSOP – Start of packet flag. It indicates the buffer associated with this descriptor is transmitted first. This flag is set by the CPU. 0 – Not end of packet. 1 – End of packet Bits 31:28–Reserved
1Ch	MC_STS	Multichannel Status bits. Bits 27:0 – Reserved Bit 28 – IE – DMA Internal Error due to under-run or over-run conditions. 0 – No DMA Internal Errors 1 – DMA Internal Error detected. DMA Engine halts. Bit 29 – SE – DMA Slave Error. This error occurs if the slave read from the Memory Map interface issues a Slave Error. 0 – No DMA Slave Errors 1 – DMA Slave Error detected. DMA Engine halts Bit 30 – DE – DMA Decode Error. This error occurs if the address request is to an invalid address. 0 – No DMA Decode Errors 1 – DMA Decode Error detected. DMA Engine halts Bit 31 – Cmp – Completed. This indicates to the software that the DMA engine has completed the transfer. 0 – Descriptor not completed 1 – Descriptor completed

Notes:

1. The ARCACHE, ARUSER values are important from the AXI read perspective. These values should be specified in the descriptor as needed. For normal operation ARCACHE should be set to 0011 while ARUSER can be set to 0000.
2. A value of 0 on VSIZE is illegal and results in the multichannel DMA not functioning as expected.

S2MM (RX) Descriptor

Figure 42: **RX Descriptor**



RX Descriptor Fields

Table 42: **RX Descriptor Fields**

Address Space Offset	Name	Description
00h	NXTDESC	Bits 5:0 – Reserved Bits 31:6 – Next Descriptor Pointer
04h	NXTDESC_MSB	Provides upper 32 bits of the next descriptor pointer. Applicable when DMA is configured for an address space greater than 32.
08h	BUFFER_ADDRESS	Bits 31:0 – Buffer Address Provides the location of the buffer space available to store data transferred from Stream to Memory Map. The address should be aligned to Memory Map data width.
0Ch	BUFFER_ADDRESS_MSB	Provides the upper 32 bits of buffer address. This is used only when AXI DMA is configured for an address space greater than 32.
10h	CACHE_USER_CTL	Bit 23:0 – Reserved
		Bit 27:24 – AWCACHE – Cache type. This signal provides additional information about the cacheable characteristics of the transfer. See the *AMBA® AXI and ACE Protocol Specification* for a different decoding mechanism. AWCACHE values from RX descriptor are presented on AWCACHE [3:0] bus during address cycle. Default value of this field should be 0011.
		Bits 31:28 – AWUSER – sideband signals used for user- defined information. AWUSER values from RX descriptor are presented on AWUSER [3:0]. AWUSER values and their interpretations are user-defined. You can keep AWUSER static for entire packet by programming same values in all the descriptors within a chain.

Table 42: **RX Descriptor Fields** *(cont'd)*

Address Space Offset	Name	Description
14h	STRIDE_VSIZE	Bits 15:0 – Stride Control. It is the address distance between the first address of successive “horizontal” writes. Writes start at the Buffer Address and write HSIZE bytes, then skip STRIDE-HSIZE addresses and write HSIZE bytes, and so on. This continues until VSIZE has been written. On AXI4-Stream this is received on the s_axis_s2mm_ interface as one contiguous packet and is terminated with a single assertion of TLAST on the last data beat of the transfer.
		Bits 18:16 – Reserved
		Bits 31:19 – Number of “horizontal lines” for stride access. Can represent two-dimensional video data or the size of a 2-D matrix. VSIZE number of transfers, each HSIZE bytes long, are expected to be received for each packet.
18h	HSIZE	Bits 15:0 – Number of bytes to transfer in each “horizontal line” from successive contiguous byte addresses. Can represent a portion of a video line or a portion of a matrix row when matrix is stored in row major order. Bits 31:16 – Reserved

Table 42: **RX Descriptor Fields** *(cont'd)*

Address Space Offset	Name	Description
1Ch	MC_STS	Multichannel Status bits. Bits 4:0 – TDEST provides routing information for the data stream. TDEST values are static for entire packet. TDEST values are captured from incoming stream and updated in this field.
		Bits 7:5 – Reserved
		Bits 12:8 – TID provides a stream identifier. TID values are static for entire packet. TID values are captured from incoming stream and updated in this field.
		Bits 15:13 – Reserved
		Bits 19:16 – TUSER – sideband signals used for user-defined information. TUSER values are static for entire packet. TUSER values are captured from incoming stream and updated in this field.
		Bits 25:20 – Reserved
		Bits 25:24 – Reserved
		Bit 26 – RXEOP – End of packet flag. It indicates the buffer associated with this descriptor contains the last part of packet. This flag is set by AXI DMA. 0 – Not end of packet 1 – End of packet
		Bit 27 – RXSOP – Start of packet flag. It indicates the buffer associated with this descriptor contains the start of the packet. This flag is set by AXI DMA. 0 – Not end of packet 1 – End of packet
		Bit 28 – IE – DMA Internal Error due to under-run or over-run conditions. 0 – No DMA Internal Errors 1 –DMA Internal Error detected. DMA Engine halts.
		Bit 29 – SE – DMA Slave Error. This error occurs if the slave read from the Memory Map interface issues a Slave Error. 0 – No DMA Slave Errors 1 –DMA Slave Error detected. DMA Engine halts.
		Bit 30 – DE – DMA Decode Error. This error occurs if the address request is to an invalid address. 0 – No DMA Decode Errors 1 –DMA Decode Error detected. DMA Engine halts.
		Bit 31 – Cmp – Completed. This indicates to the software that the DMA engine has completed the transfer. 0 – Descriptor not completed 1 –Descriptor completed

Notes:

1. The AWCACHE, AWUSER values are important from the AXI write prospective. These values should be specified in the descriptor as needed. For normal operation AWCACHE should be set to 0011 while AWUSER can be set to 0000.
2. A value of '0' on VSIZE is illegal and results in the multichannel DMA not functioning as expected.

AXI DMA Multichannel Operation

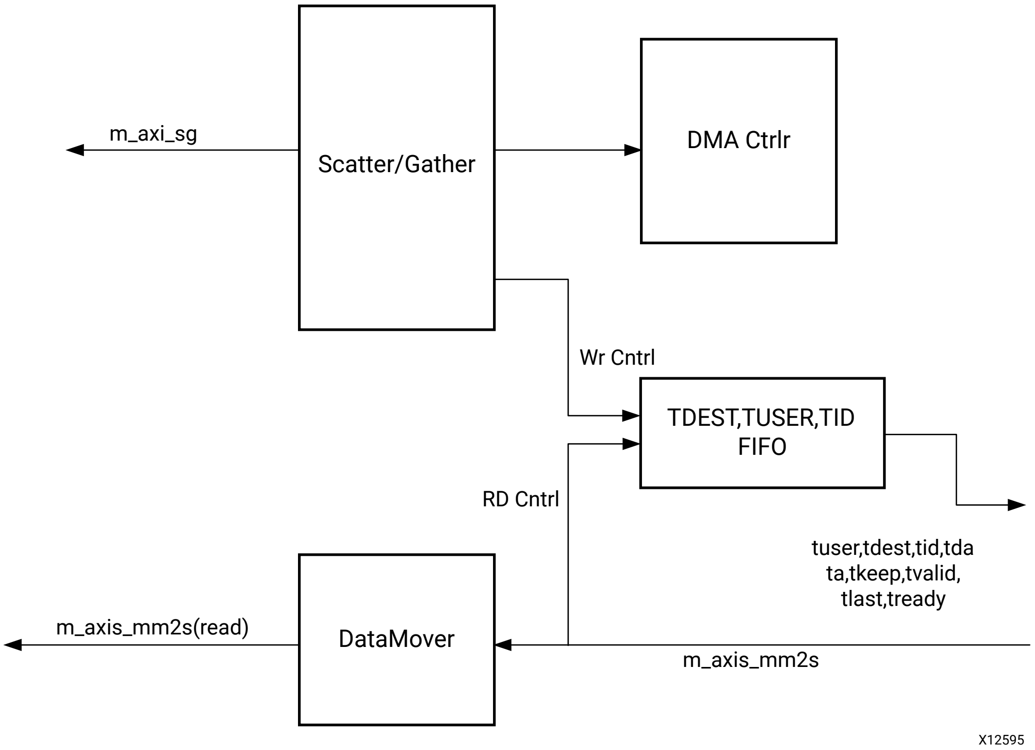
This section describes the end-to-end control and data flow of descriptors and associated data for both MM2S side and S2MM side.

MM2S

MM2S is similar to normal AXI DMA operation. When MM2S_CURDESC and MM2S_TAILDESC are programmed by software, AXI DMA fetches a chain of descriptors and processes until it reaches tail descriptor. In AXI DMA, TDEST, TID, and TUSER fields are assumed to remain constant for an entire packet as defined in the descriptors. That is, each packet transfer across a logical channel defined by (TDEST, TID, TUSER) runs to completion before the DMA transfers another packet. Although packet transfers for multiple channels can be interleaved, after started, each must run to completion before another transfer can occur. It is your responsibility to avoid deadlock scenarios under this assumption. The AXI DMA does not signal error conditions if the (TDEST, TID, TUSER) fields within the descriptors do not adhere to these assumptions. It is up to the software to maintain consistency.

TX Descriptor contains control and status fields in multichannel mode. There is no status update on this descriptor for the number of bytes transferred at the end of the transaction. Error information is captured in registers along with the current descriptor pointer of the failing descriptor. The completion of a transfer for a chain can be known by polling the IDLE bit of MM2S_DMASR or through an interrupt by enabling it in MM2S_DMACR.

Figure 43: **MM2S Control and Datapath Flow**

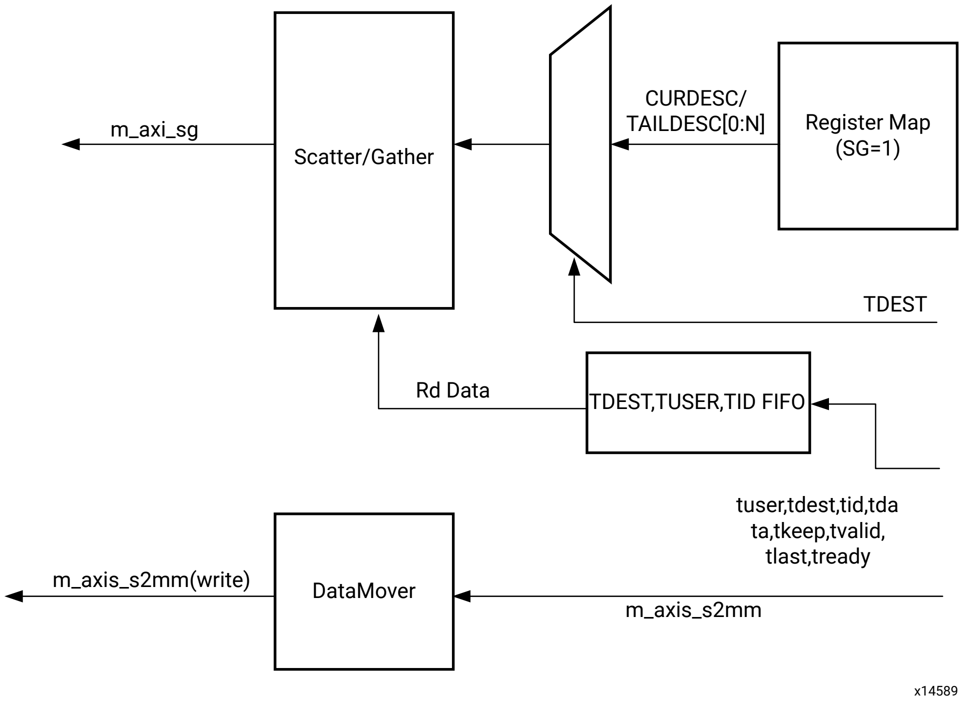


S2MM

The `tdest` signals are sampled from the incoming stream. This value is used by the DMA controller to read the corresponding S2MM_CURDESC and S2MM_TAILDESC and present them to the Scatter Gather module. In turn, this SG module fetches a chain of descriptors for that `tdest`. This means that the Buffer Descriptor chain setup should be completed (for all the channels) before the packet arrives on the S2MM channel. The Current Descriptor and Tail Descriptor registers should be programmed before the packet/data arrives. AXI DMA holds the streaming data by deasserting `tready` until the corresponding descriptors are fetched. Then, AXI DMA writes start at the Buffer Address and continues until `tlast` is received from the streaming side. For a particular channel when the data transfer is completed, the current descriptor register will be loaded with the address of the next descriptor in that chain. This is to ensure that the next packet is serviced correctly. You must ensure that the descriptor chain is setup correctly and Tail Descriptor is handled properly to avoid jumping of Current Descriptor.

TDEST, TID, and TUSER values are captured from incoming streams and stored internally. These values are not expected to change in the middle of a packet. These values are updated in each descriptor of the chain after the completion of data writes for that descriptor onto the Memory Map side along with other status bits.

Figure 44: **S2MM Control and Datapath Flow**



2-D Transfers

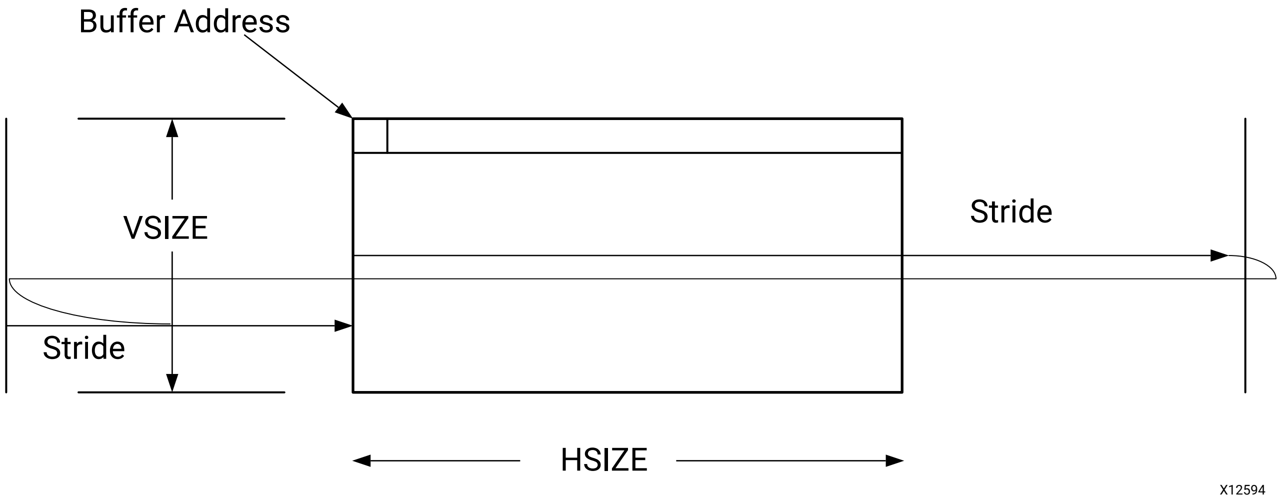
In Multichannel Mode, AXI DMA supports 2-D memory access patterns to be efficiently transferred with an AXI4-Stream channel.

Access patterns are controlled with descriptor fields HSIZE, VSIZE, and STRIDE, which enable the transfer of sub-blocks within the (implicit) 2-D array. HSIZE is specified between starting addresses for successive 'row' sub-blocks. For 2-D transfers, the HSIZE, VISZE and STRIDE should be byte aligned. Having unaligned values of HSIZE, VSIZE or STRIDE causes unexpected behavior.

Each read (MM2S) or write (S2MM) transfer consists of VSIZE transfers, each of size HSIZE. The starting address of each successive transfer is STRIDE address from the starting address of the previous transfer (initially, the BaseAddr of the packet transfer).

The following figure shows the example of the two-dimensional data format.

Figure 45: **2-D Data Format**



MM2S

Reads start at the Buffer Address and consists of VSIZE read bursts, each having HSIZE bytes. The starting address of each read burst is a STRIDE address greater than the starting address of the previous burst read.

Example: Buffer Address = 08, VSIZE = 06, HSIZE = 256 bytes and Stride = 512 bytes

In this case, Reads start at buffer address location 08 and continue to read HSIZE (256) bytes. The second line starts at Buffer address+Stride = 512+8 = 520. It continues to read HSIZE (256) bytes. The third line starts at 520+512 = 1,032 and the fourth line starts at 1032+512 = 1,544. Reads continue in this pattern for VSIZE lines.

On AXI4-Stream this is transmitted on the `m_axis_mm2s` interface as one contiguous packet and is terminated with the assertion of `tlast` on the last data beat of the transfer.

S2MM

Writes start at the Buffer Address and consist of VSIZE write bursts, each having HSIZE bytes. The starting address of each write burst is a STRIDE address greater than the starting address of the previous burst write. On the AXI4-Stream interface, this is received on the `s_axis_s2mm_interface` as one contiguous packet and is terminated with a single assertion of `tlast` on the last data beat of the transfer. The size of the arriving packet should match with what is programmed in Buffer Descriptors.

Limitations with Multichannel Mode

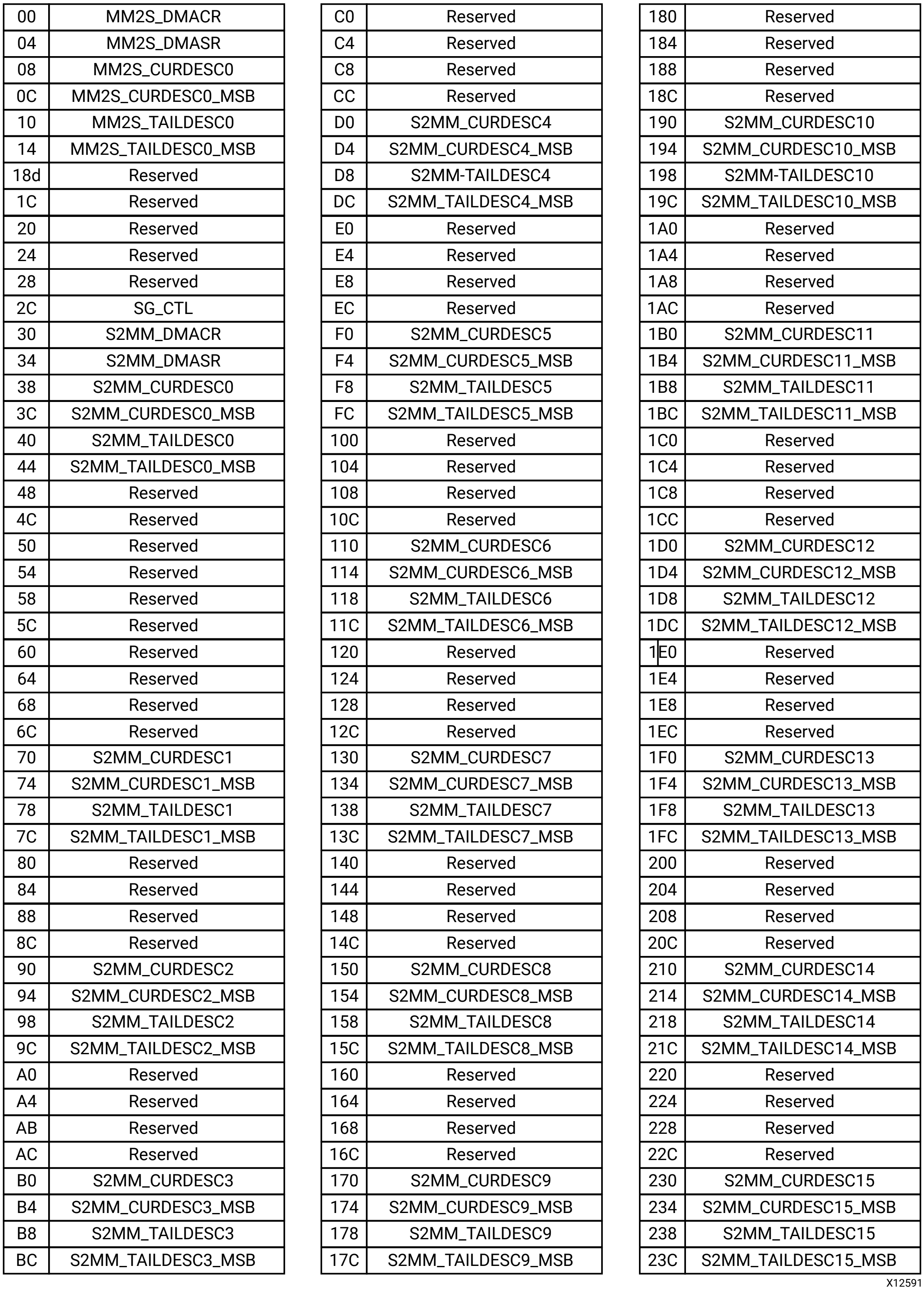
- Does not support descriptor queuing in S2MM path for multichannel mode

- Does not support small packet sizes for S2MM path (back-to-back packets of 4 or less data beats)
- For a 2-D access the address and the Hsize have to be aligned to the Memory Map data width.

Register Map for Multichannel (SG = 1)

Register map is modified to support up to 16 TDEST on the S2MM path.

Figure 46: **Register Map for Multichannel Support in SG = 1**



00	MM2S_DMACR
04	MM2S_DMASR
08	MM2S_CURDESC0
0C	MM2S_CURDESC0_MSB
10	MM2S_TAILDESC0
14	MM2S_TAILDESC0_MSB
18d	Reserved
1C	Reserved
20	Reserved
24	Reserved
28	Reserved
2C	SG_CTL
30	S2MM_DMACR
34	S2MM_DMASR
38	S2MM_CURDESC0
3C	S2MM_CURDESC0_MSB
40	S2MM_TAILDESC0
44	S2MM_TAILDESC0_MSB
48	Reserved
4C	Reserved
50	Reserved
54	Reserved
58	Reserved
5C	Reserved
60	Reserved
64	Reserved
68	Reserved
6C	Reserved
70	S2MM_CURDESC1
74	S2MM_CURDESC1_MSB
78	S2MM_TAILDESC1
7C	S2MM_TAILDESC1_MSB
80	Reserved
84	Reserved
88	Reserved
8C	Reserved
90	S2MM_CURDESC2
94	S2MM_CURDESC2_MSB
98	S2MM_TAILDESC2
9C	S2MM_TAILDESC2_MSB
A0	Reserved
A4	Reserved
AB	Reserved
AC	Reserved
B0	S2MM_CURDESC3
B4	S2MM_CURDESC3_MSB
B8	S2MM_TAILDESC3
BC	S2MM_TAILDESC3_MSB
C0	Reserved
C4	Reserved
C8	Reserved
CC	Reserved
D0	S2MM_CURDESC4
D4	S2MM_CURDESC4_MSB
D8	S2MM-TAILDESC4
DC	S2MM_TAILDESC4_MSB
E0	Reserved
E4	Reserved
E8	Reserved
EC	Reserved
F0	S2MM_CURDESC5
F4	S2MM_CURDESC5_MSB
F8	S2MM_TAILDESC5
FC	S2MM_TAILDESC5_MSB
100	Reserved
104	Reserved
108	Reserved
10C	Reserved
110	S2MM_CURDESC6
114	S2MM_CURDESC6_MSB
118	S2MM_TAILDESC6
11C	S2MM_TAILDESC6_MSB
120	Reserved
124	Reserved
128	Reserved
12C	Reserved
130	S2MM_CURDESC7
134	S2MM_CURDESC7_MSB
138	S2MM_TAILDESC7
13C	S2MM_TAILDESC7_MSB
140	Reserved
144	Reserved
148	Reserved
14C	Reserved
150	S2MM_CURDESC8
154	S2MM_CURDESC8_MSB
158	S2MM_TAILDESC8
15C	S2MM_TAILDESC8_MSB
160	Reserved
164	Reserved
168	Reserved
16C	Reserved
170	S2MM_CURDESC9
174	S2MM_CURDESC9_MSB
178	S2MM_TAILDESC9
17C	S2MM_TAILDESC9_MSB
180	Reserved
184	Reserved
188	Reserved
18C	Reserved
190	S2MM_CURDESC10
194	S2MM_CURDESC10_MSB
198	S2MM-TAILDESC10
19C	S2MM_TAILDESC10_MSB
1A0	Reserved
1A4	Reserved
1A8	Reserved
1AC	Reserved
1B0	S2MM_CURDESC11
1B4	S2MM_CURDESC11_MSB
1B8	S2MM_TAILDESC11
1BC	S2MM_TAILDESC11_MSB
1C0	Reserved
1C4	Reserved
1C8	Reserved
1CC	Reserved
1D0	S2MM_CURDESC12
1D4	S2MM_CURDESC12_MSB
1D8	S2MM_TAILDESC12
1DC	S2MM_TAILDESC12_MSB
1E0	Reserved
1E4	Reserved
1E8	Reserved
1EC	Reserved
1F0	S2MM_CURDESC13
1F4	S2MM_CURDESC13_MSB
1F8	S2MM_TAILDESC13
1FC	S2MM_TAILDESC13_MSB
200	Reserved
204	Reserved
208	Reserved
20C	Reserved
210	S2MM_CURDESC14
214	S2MM_CURDESC14_MSB
218	S2MM_TAILDESC14
21C	S2MM_TAILDESC14_MSB
220	Reserved
224	Reserved
228	Reserved
22C	Reserved
230	S2MM_CURDESC15
234	S2MM_CURDESC15_MSB
238	S2MM_TAILDESC15
23C	S2MM_TAILDESC15_MSB

X12591

Chapter 4

Designing with the Core

This section includes guidelines and additional information to facilitate designing with the core.

General Design Guidelines

Use the Example Design

Each instance of the AXI DMA core created by the Vivado design tool is delivered with an example design that can be implemented in a device and then simulated. This design can be used as a starting point for your own design or can be used to sanity-check your application in the event of difficulty. See the Example Design content for information about using and customizing the example designs for the core.

Registering Signals

To simplify timing and increase system performance in a programmable device design, keep all inputs and outputs registered between the user application and the core. This means that all inputs and outputs from the user application should come from, or connect to, a flip-flop. While registering signals might not be possible for all paths, it simplifies timing analysis and makes it easier for the AMD tools to place and route the design.

Recognize Timing Critical Signals

The constraints provided with the example design identify the critical signals and timing constraints that should be applied.

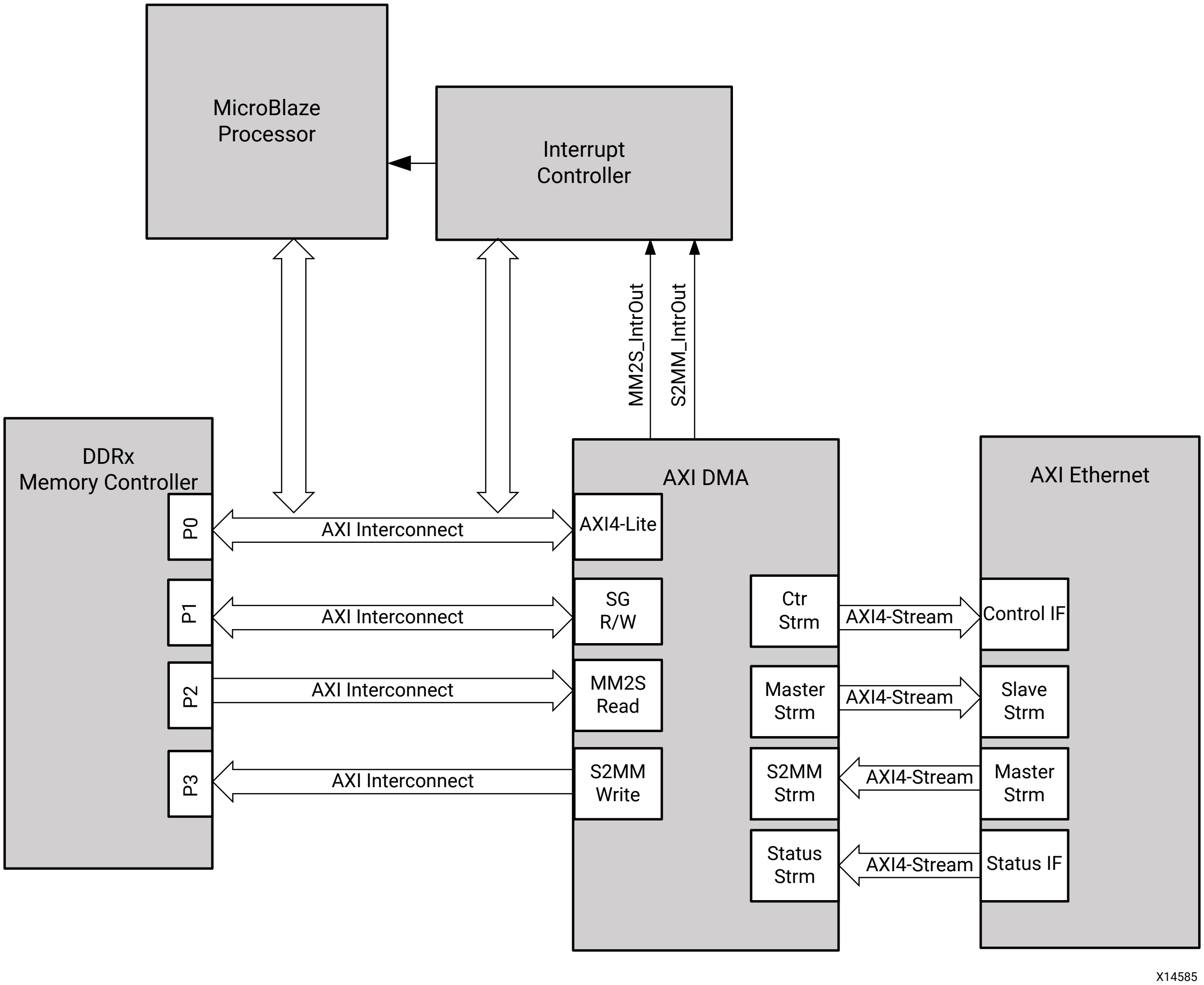
Make Only Allowed Modifications

You should not modify the core. Any modifications can have adverse effects on system timing and protocol compliance. Supported user configurations of the core can only be made by selecting the options in the customization IP dialog box when the core is generated.

Typical System Interconnect

The AXI DMA core is designed to be connected through the AXI Interconnect in the user system. A typical MicroBlaze™ processor configuration is shown in the following figure. The system microprocessor has access to the AXI DMA through the AXI4-Lite interface. An integrated Scatter/Gather Engine fetches buffer descriptors from system memory which then coordinates primary data transfers between AXI IP and DDRx. Optional control and status streams provide packet-associated information, such as checksum offload control/status, to and from an Ethernet based IP. The dual interrupt output of the AXI DMA core is routed to the System Interrupt Controller.

Figure 47: **Typical MicroBlaze Processor System Configuration (AXI Ethernet)**



The AXI DMA core can also be connected to a user system other than with an Ethernet-based AXI IP. Control and status streams are optional and can be used with Ethernet based IP cores only.

Note**:** In the absence of any setup (that is, before it is programmed to run), AXI DMA will pull the `s_axis_s2mm_tready` signal Low after taking in four beats of streaming data. This will throttle the input data stream. To have a minimum amount of throttling, ensure that the AXI DMA is set up to run much before the actual data arrives.

Clocking

There are four clock inputs:

- `m_axi_mm2s_aclk` for MM2S interface
- `m_axi_s2mm_aclk` for S2MM interface
- `s_axi_lite_aclk` for AXI4-Lite control interface
- `m_axi_sg_clk` for Scatter Gather Interface

AXI DMA provides two clocking modes of operation: asynchronous and synchronous. Setting Enable Asynchronous Clocks enables asynchronous mode and creates four clock domains. This allows high-performance users to run the primary datapaths at a higher clock rate than the DMA control (for example, AXI4-Lite interface, SG Engine, DMA Controller) helping in FPGA placement and timing.

In synchronous mode, all logic runs in a single clock domain. `m_axi_sg_aclk`, `m_axi_mm2s_aclk`, and `m_axi_s2mm_aclk` must be tied to the same source, the `s_axi_lite_aclk` can be connected to a slower clock. In asynchronous mode, clocks can be run asynchronously, however `s_axi_lite_aclk` must be less than or equal to `m_axi_sg_aclk` and `m_axi_sg_aclk` must be less than or equal to the slower of `m_axi_mm2s_aclk` or `m_axi_s2mm_aclk`.

The relationship between signal sets and their corresponding clocks in asynchronous mode is shown in the following table.

Table 43: **Asynchronous Clock Distribution**

Clock Source	I/O Ports (Scatter Gather Enabled)	I/O Ports (Scatter Gather Disabled)
s_axi_lite_aclk	All s_axi_lite_* Signals mm2s_introut s2mm_introut axi_resetn	All s_axi_lite_* Signals mm2s_introut s2mm_introut axi_resetn
m_axi_sg_aclk	Allm_axi_sg_* Signals	N/A

Table 43: **Asynchronous Clock Distribution** *(cont'd)*

Clock Source	I/O Ports (Scatter Gather Enabled)	I/O Ports (Scatter Gather Disabled)
m_axi_mm2s_aclk	All m_axi_mm2s_* Signals All m_axis_mm2s_* Signals mm2s_prmry_reset_out_n mm2s_cntrl_reset_out_n	All m_axi_mm2s_* Signals All m_axis_mm2s_* Signals mm2s_prmry_reset_out_n
m_axi_s2mm_aclk	All m_axi_s2mm_* Signals All s_axis_s2mm_* Signals s2mm_prmry_reset_out_n s2mm_sts_reset_out_n	All m_axi_s2mm_* Signals All s_axis_s2mm_* Signals s2mm_prmry_reset_out_n

Resets

The `axi_resetn` signal needs to be asserted a minimum of 16 of the slowest clock cycles and needs to be synchronized to `s_axi_lite_aclk`.

Programming Sequence

Direct Register Mode (Simple DMA)

Direct Register Mode (Scatter Gather Engine is disabled) provides a configuration for doing simple DMA transfers on MM2S and S2MM channels that requires less FPGA resource utilization. Transfers are initiated by accessing the DMACR, the Source or Destination Address and the Length registers. When the transfer is completed, a DMASR.IOC_Irq asserts for the associated channel and if enabled generates an interrupt out.

A DMA operation for the MM2S channel is set up and started by the following sequence:

1. Start the MM2S channel running by setting the run/stop bit to 1 (MM2S_DMACR.RS = 1). The halted bit (DMASR.Halted) should deassert indicating the MM2S channel is running.
2. If desired, enable interrupts by writing a 1 to MM2S_DMACR.IOC_IrqEn and MM2S_DMACR.Err_IrqEn. The delay interrupt, delay count, and threshold count are not used when the AXI DMA is configured for Direct Register Mode.
3. Write a valid source address to the MM2S_SA register. If AXI DMA is configured for an address space greater than 32, then program the MM2S_SA MSB register. If the AXI DMA is not configured for Data Re-Alignment, then a valid address must be aligned or undefined results occur. What is considered aligned or unaligned is based on the stream data width. When AXI_DMA is configured in Micro Mode, it is your responsibility to specify the correct address. Micro DMA does not take care of the 4K boundary.

For example, if Memory Map Data Width = 32, data is aligned if it is located at word offsets (32-bit offset), that is 0x0, 0x4, 0x8, 0xC, and so forth. If DRE is enabled and Streaming Data Width < 128, then the Source Addresses can be of any byte offset.

4. Write the number of bytes to transfer in the MM2S_LENGTH register. A value of zero written has no effect. A non-zero value causes the MM2S_LENGTH number of bytes to be read on the MM2S AXI4 interface and transmitted out of the MM2S AXI4-Stream interface. The MM2S_LENGTH register must be written last. All other MM2S registers can be written in any order. In the case of Micro DMA, this value cannot exceed [Burst_length * (Memory Mapped Data Width)/8].

A DMA operation for the S2MM channel is set up and started by the following sequence:

1. Start the S2MM channel running by setting the run/stop bit to 1 (S2MM_DMACR.RS = 1). The halted bit (DMASR.Halted) should deassert indicating the S2MM channel is running.
2. If desired, enable interrupts by writing a 1 to S2MM_DMACR.IOC_IrqEn and S2MM_DMACR.Err_IrqEn. The delay interrupt, delay count, and threshold count are not used when the AXI DMA is configured for Direct Register Mode.
3. Write a valid destination address to the S2MM_DA register. If AXI DMA is configured for an address space greater than 32, program the S2MM_DA MSB register.
4. If the AXI DMA is not configured for Data Re-Alignment then a valid address must be aligned or undefined results occur. What is considered aligned or unaligned is based on the stream data width.

   For example, if Memory Map Data Width= 32, data is aligned if it is located at word offsets (32-bit offset), that is, 0x0, 0x4, 0x8, 0xC, and so forth. If DRE is enabled and Streaming Data Width < 128 then the Destination Addresses can be of any byte offset.
5. Write the length in bytes of the receive buffer in the S2MM_LENGTH register. A value of zero has no effect. A non-zero value causes a write on the S2MM AXI4 interface of the number of bytes received on the S2MM AXI4-Stream interface. A value greater than or equal to the largest received packet must be written to S2MM_LENGTH. A receive buffer length value that is less than the number of bytes received produces undefined results. When AXI DMA is configured in Micro mode, this value should exactly match the bytes received on the S2MM AXI4-Stream interface. The S2MM_LENGTH register must be written last. All other S2MM registers can be written in any order.

Scatter/Gather Mode

AXI DMA operation requires a memory-resident data structure that holds the list of DMA operations to be performed. This list of instructions is organized into what is referred to as a descriptor chain. Each descriptor has a pointer to the next descriptor to be processed. The last descriptor in the chain then points back to the first descriptor in the chain.

Scatter Gather operation allows a packet to be described by more than one descriptor. A typical use for this feature is to allow storing or fetching of headers from a location in memory and payload data from another location. Software applications that take advantage of this can improve throughput. To delineate packets in a buffer descriptor chain, the Start of Frame bit (TXSOF) and End of Frame bit (TXEOF) are used. When the DMA fetches a descriptor with the TXSOF bit set, the start of a packet is triggered. The packet continues with fetching the subsequent descriptors until it fetches a descriptor with the TXEOF bit set.

On the receive (S2MM) channel when a packet starts to be received, the AXI DMA marks the descriptor with an RXSOF indicating to the software that the data buffer associated with this descriptor contains the beginning of a packet. If the packet being received is longer in byte count than what was specified in the descriptor, the next descriptor buffer is used to store the remainder of the receive packet. This fetching and storing process continues until the entire receive packet has been transferred. The descriptor being processed when the end of the packet is received is marked by AXI DMA with an RXEOF=1. This indicates to the software that the buffer associated with this descriptor contains the end of the packet.

The status field of each descriptor contains the number of bytes actually transferred for that particular descriptor. The software can determine the total number of bytes transferred for the receive packet by walking from the RXSOF descriptor through the descriptor chain to the RXEOF descriptor. The Scatter Gather continues to fetch one extra descriptor and store. This process improves the DMA performance to a great extent.

Scatter Gather operations begin with the setting up of control registers and descriptor pointers.

A DMA operation for the MM2S channel is set up and started by using the following sequence:

1. Write the address of the starting descriptor to the Current Descriptor register. If AXI DMA is configured for an address space greater than 32, then also program the MSB 32 bits of the current descriptor.
2. Start the MM2S channel running by setting the run/stop bit to 1 (MM2S_DMACR.RS =1). The Halted bit (DMASR.Halted) should deassert indicating the MM2S channel is running.
3. If desired, enable interrupts by writing a 1 to MM2S_DMACR.IOC_IrqEn and MM2S_DMACR.Err_IrqEn.
4. Write a valid address to the Tail Descriptor register. If AXI DMA is configured for an address space greater than 32, then also program the MSB 32 bits of the tail descriptor.
5. Writing to the Tail Descriptor register triggers the DMA to start fetching the descriptors from the memory. In case of multichannel configuration, the fetching of descriptors starts when the packet arrives on the S2MM channel.
6. The fetched descriptors are processed, Data is read from the memory and then output to the MM2S streaming channel.

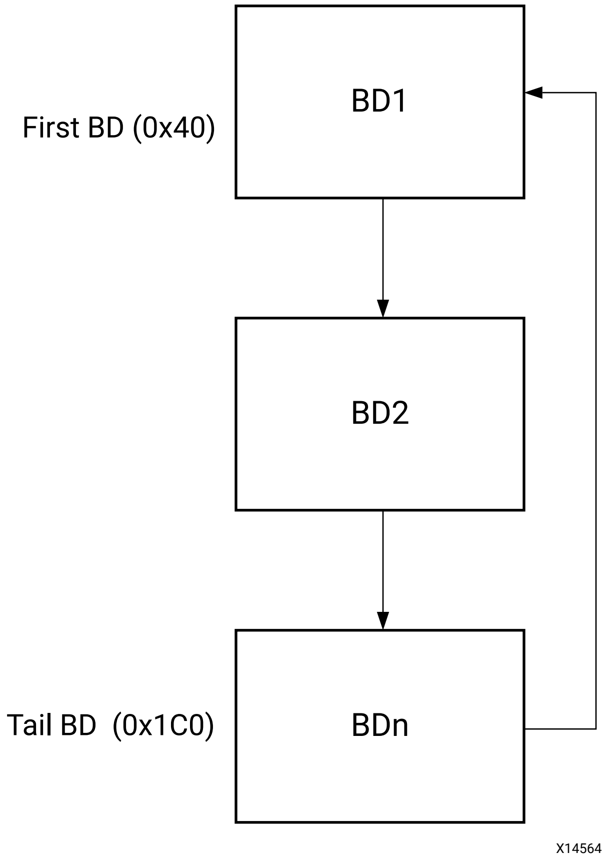
A DMA operation for the S2MM channel is set up and started by using the following sequence:

1. Write the address of the starting descriptor to the Current Descriptor register. If AXI DMA is configured for an address space greater than 32, then also program the MSB 32 bits of the current descriptor.
2. Start the S2MM channel running by setting the run/stop bit to 1 (S2MM_DMACR.RS =1).
3. The halted bit (DMASR.Halted) should deassert indicating the S2MM channel is running.
4. If desired, enable interrupts by writing a 1 to S2MM_DMACR.IOC_IrqEn and S2MM_DMACR.Err_IrqEn.
5. Write a valid address to the Tail Descriptor register. If AXI DMA is configured for an address space greater than 32, then also program the MSB 32 bits of the current descriptor.
6. Writing to the Tail Descriptor register triggers the DMA to start fetching the descriptors from the memory. The fetched descriptors are processed and any data received from the S2MM streaming channel is written to the memory.

Cyclic DMA Mode

AXI DMA can be run in cyclic mode by making certain changes to the Buffer Descriptor (BD) chain setup. In cyclic mode, DMA fetches and processes the same BDs without interruption. The DMA continues to fetch and process until it is stopped or reset. To enable cyclic operation, the BD chain should be set up as shown in the following figure.

Figure 48: **BD Chain**



In this setup the Tail BD points back to the first BD. The Tail Descriptor register does not serve any purpose and is used only to trigger the DMA engine. Follow the same programming sequences as mentioned in Scatter/Gather Mode. Ensure that the cyclic bit in the control register is set.

Program the Tail Descriptor register with some value which is not a part of the BD chain. Say for example, `0x50`.

After the Tail Descriptor register is programmed, the DMA starts fetching and processing the BDs (which are set up in a ring fashion) until the DMA is stopped or reset.

Chapter 5

Design Flow Steps

This section describes customizing and generating the core, constraining the core, and the simulation, synthesis, and implementation steps that are specific to this IP core. More detailed information about the standard AMD Vivado™ design flows and the IP integrator can be found in the following Vivado Design Suite user guides:

- *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
- *Vivado Design Suite User Guide: Designing with IP* (UG896)
- *Vivado Design Suite User Guide: Getting Started* (UG910)
- *Vivado Design Suite User Guide: Logic Simulation* (UG900)

Customizing and Generating the Core

The AXI DMA can be found in the following places in the Vivado IP catalog.

- AXI_Infrastructure, Communication_&_Networking\Ethernet
- Embedded_Processing\AXI_Infrastructure\DMA

To access the AXI DMA, do the following:

1. Open an existing project or create a new project using the Vivado design tools.
2. Open the IP catalog and navigate to any of the taxonomies.
3. Double-click **AXI Direct Memory Access** to bring up the AXI DMA Customize IP window.

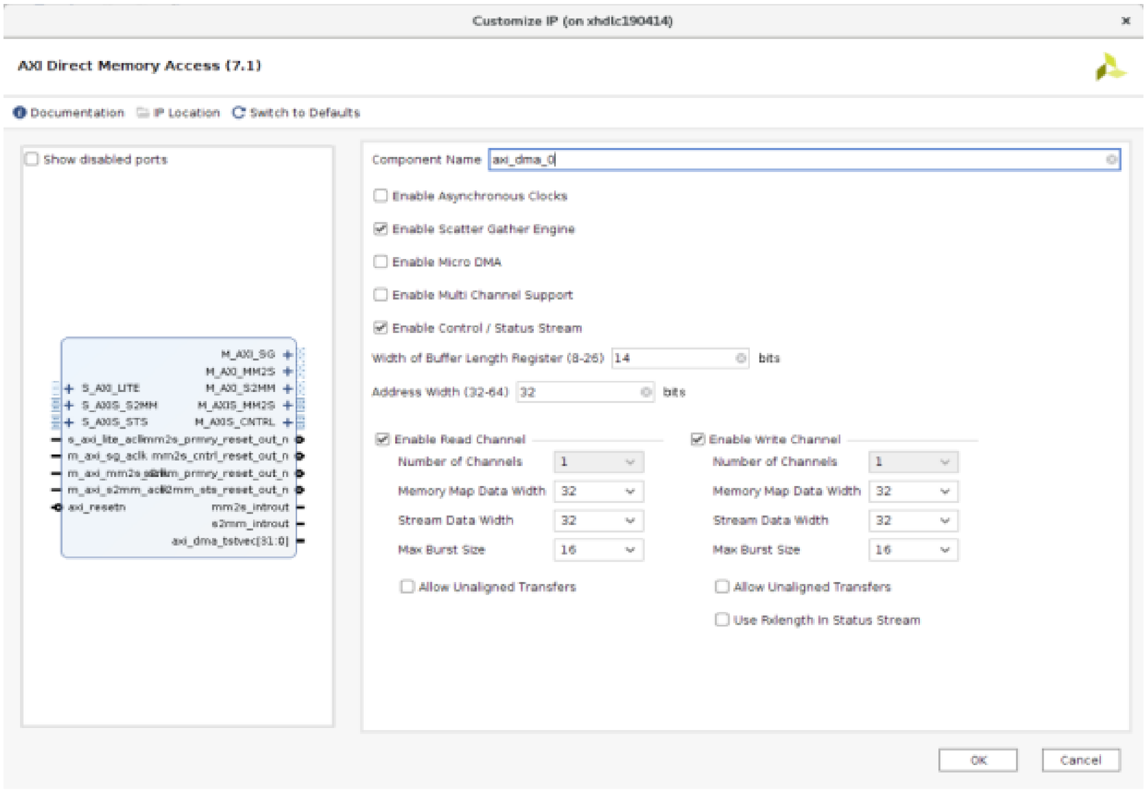
If you are customizing and generating the core in the Vivado IP integrator, see the *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) for detailed information. IP integrator might auto-compute certain configuration values when validating or generating the design. To check whether the values do change, see the description of the parameter in this chapter. To view the parameter value, run the `validate_bd_design` command in the Tcl console.

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) and the *Vivado Design Suite User Guide: Getting Started* (UG910).

Figures in this chapter are illustrations of the Vivado IDE. The layout depicted here might vary from the current version.

Field Descriptions

Figure 49: **Customize IP Options**



Component Name axi_dma_0

□ Enable Asynchronous Clocks

☑ Enable Scatter Gather Engine

□ Enable Micro DMA

□ Enable Multi Channel Support

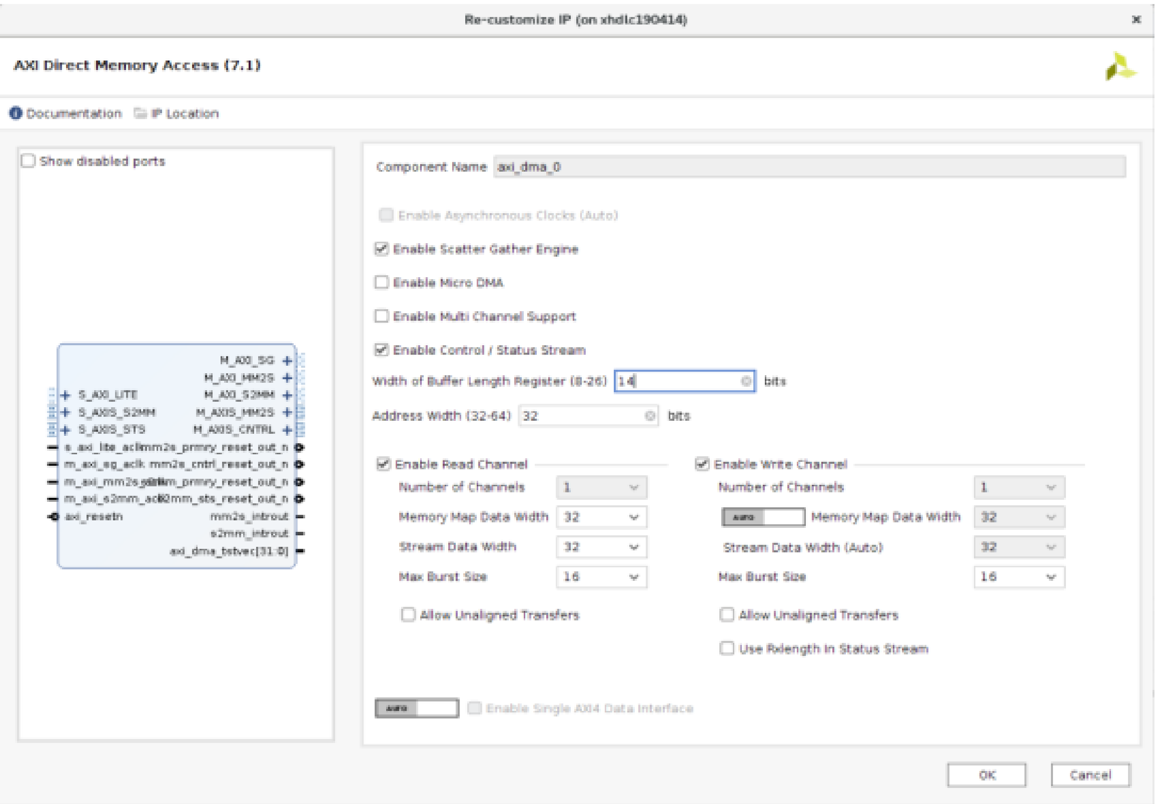
☑ Enable Control / Status Stream

Width of Buffer Length Register (8-26) 14 bits

Address Width (32-64) 32 bits

☑ Enable Read Channel		☑ Enable Write Channel	
Number of Channels	1	Number of Channels	1
Memory Map Data Width	32	Memory Map Data Width	32
Stream Data Width	32	Stream Data Width	32
Max Burst Size	16	Max Burst Size	16
□ Allow Unaligned Transfers		□ Allow Unaligned Transfers	
		□ Use Rxlength In Status Stream	

Figure 50: **IP Integrator**



Re-customize IP (on xhdlc190414)

AXI Direct Memory Access (7.1)

Documentation · IP Location

□ Show disabled ports

Component Name: axi_dma_0

□ Enable Asynchronous Clocks (Auto)

☑ Enable Scatter Gather Engine

□ Enable Micro DMA

□ Enable Multi Channel Support

☑ Enable Control / Status Stream

Width of Buffer Length Register (8-26): 14 bits

Address Width (32-64): 32 bits

☑ Enable Read Channel

- Number of Channels: 1
- Memory Map Data Width: 32
- Stream Data Width: 32
- Max Burst Size: 16
- □ Allow Unaligned Transfers

☑ Enable Write Channel

- Number of Channels: 1
- Memory Map Data Width: 32
- Stream Data Width (Auto): 32
- Max Burst Size: 16
- □ Allow Unaligned Transfers
- □ Use Rxlength In Status Stream

□ Enable Single AXI4 Data Interface

- **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 "_".

- **Enable Asynchronous Clocks:** This setting provides the ability to operate the MM2S interface `m_axi_mm2s_aclk`, S2MM interface `m_axi_s2mm_aclk`, AXI4-Lite control interface `s_axi_lite_aclk`, and the Scatter Gather Interface `m_axi_sg_aclk` asynchronously from each other. When Asynchronous Clocks are enabled, the frequency of `s_axi_lite_aclk` must be less than or equal to `m_axi_sg_aclk`. Likewise `m_axi_sg_aclk` must be less than or equal to the slower of `m_axi_mm2s_aclk` and `m_axi_s2mm_aclk`. When in synchronous mode, all clocks inputs should be connected to the same clock signal. This parameter is automatically set in the Vivado IP integrator based on the clocks connected to axi_dma.

  > **TIP:** *This parameter is automatically set when the IP is used in the Vivado IP integrator.*

- **Enable Scatter Gather Engine:** Checking this option enables Scatter Gather Mode operation and includes the Scatter Gather Engine in AXI DMA. Unchecking this option enables Direct Register Mode operation, excluding the Scatter Gather Engine from AXI DMA. Disabling the Scatter Gather Engine causes all output ports for the Scatter/Gather engine to be tied to zero and all of the input ports to be left open.

- **Enable Micro DMA:** Checking this option generates a highly optimized DMA which is low on resource count. This setting can be used for applications that transfer a very small amount of data. Program the DMA based on the configuration selected. For example, the maximum bytes that can be transferred per transaction or per BD cannot exceed the following:

  MMapData_width * Burst_length/8.

  Similarly, the 4K boundary check is also not implemented in this mode restricting addressing to burst boundaries.

- **Width of Buffer Length Register:** This integer value specifies the number of valid bits used for the Control field buffer length and Status field bytes transferred in the Scatter/Gather descriptors. It also specifies the number of valid bits in the RX Length of the Status Stream App4 field when Use Rxlength is enabled. For Direct Register Mode, it specifies the number of valid bits in the MM2S_LENGTH and S2MM_LENGTH registers. The length width directly correlates to the number of bytes being specified in a Scatter/Gather descriptor or number of bytes being specified in App4.RxLength, MM2S_LENGTH, or S2MM_LENGTH. The number of bytes is equal to $2^{\text{LengthWidth}}$ -1. So a Length Width of 26 gives a byte count of 67,108,863 bytes. This value should be set to 23 for Multichannel mode.

- **Address Width (32 - 64):** Specify the width of the Address Space. It can be any value between 32 and 64.

- **Enable MultiChannel DMA:**

  Note: The MultiChannel feature will be discontinued soon. For information on MultiChannel, see the *AXI Multichannel Direct Memory Access LogiCORE IP Product Guide* (PG288).

  Checking this option enables multichannel capability of DMA and lets you choose the number of channels for both MM2S and S2MM channels. See Multichannel DMA Support for details.

- **Enable Control/Status Stream:** Checking this option enables the AXI4 Control and Status Streams. The AXI4 Control stream allows user application metadata associated with the MM2S channel to be transmitted to a target IP. User application fields 0 through 4 of an MM2S Scatter/Gather Start Of Frame (SOF) descriptor Transmit Start Of Frame (TXSOF =1) are transmitted on the `m_axis_mm2s_cntrl` stream interface along with an associated packet being transmitted on the `m_axis_mm2s` stream interface. The AXI4 Status stream allows user application metadata associated with the S2MM channel to be received from a target IP. The received status packet populates user application fields 0 to 4 of an S2MM Scatter / Gather End of Frame (EOF) descriptor. That is the descriptor associated with the end of packet. This condition is indicated by a Receive End of Frame (RXEOF = 1) in the status word of the updated descriptor.

- **Enable Read Channel Options:** The following options affect only the MM2S Channel of the AXI Direct Memory Access (DMA) core.

  - **Enable Channel:** This option enables or disables the MM2S Channel. Enabling the MM2S Channel allows read transfers from memory to AXI4-Stream to occur. Disabling the MM2S Channel excludes the logic from the AXI DMA core. Outputs for MM2S channel are tied to zero and inputs are ignored by AXI DMA.

- **Number of Channels:** This option specifies the number of channels from 1 to 16.
- **Memory Map Data Width:** Data width in bits of the AXI MM2S Memory Map Read data bus. Valid values are 32, 64, 128,256, 512, and, 1,024.
- **Stream Data Width:** Data width in bits of the AXI MM2S AXI4-Stream Data bus. This value must be equal or less than the Memory Map Data Width. Valid values are 8, 16, 32, 64, 128, 512, and, 1,024.
- **Max Burst Size:** Burst partition granularity setting. This setting specifies the maximum size of the burst cycles on the AXI4 side of MM2S. Valid values are 2, 4, 8,16, 32, 64, 128, and 256.
- **Allow Unaligned Transfers:** 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 memory. 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.

  ***Note*:** If DRE is disabled for the respective channel, unaligned Buffer, Source, or Destination Addresses are not supported. Having an unaligned address with DRE disabled produces undefined results. DRE Support is only available for the AXI4-Stream data width setting of 512-bits and under.

- **Enable Write Channel Options:** These options affect only the S2MM Channel of the AXI DMA core.
  - **Enable Channel:** This setting enables or disables the S2MM Channel. Enabling the S2MM Channel allows write transfers from AXI4-Stream to memory to occur. Disabling the S2MM Channel excludes the logic from the AXI DMA core. Outputs for S2MM channel are tied to zero and inputs are ignored by AXI DMA.
  - **Number of Channels:** This option enables you to choose a number of channels from 1 to 16.
  - **Memory Map Data Width:** Data width in bits of the AXI S2MM Memory Map Write data bus. Valid values are 32, 64, 128, 256, 512 and, 1,024.

    TIP: *In the Vivado IP integrator, this parameter is automatically set based on the data width of the Streaming Interface. Update this parameter by changing the switch to 'Manual'.*

  - **Stream Data Width:** Data width in bits of the AXI S2MM AXI4-Stream Data bus. This value must be equal or less than the Memory Map Data Width. Valid values are 8, 16, 32, 64, 128, 512 and, 1,024.

    TIP: *When IP is used in the Vivado IP integrator, this parameter is automatically set based on the connection made to the s_axis_s2mm interface.*

  - **Max Burst Size:** This setting specifies the maximum size of the burst cycles on the AXI4 side of the S2MM channel. In other words, this setting specifies the granularity of burst mapping. Valid values are 2, 4, 8, 16, 32, 64, 128, and 256.

- **Allow Unaligned Transfers:** 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 memory. If the DRE is enabled, data writes can start from any Buffer Address byte offset, and the write data is aligned such that the first valid byte received on S2MM AXI4-Stream is written to the specified unaligned address offset.

  Note: If DRE is disabled for the respective channel, unaligned Buffer, Source, or Destination Addresses are not supported. Having an unaligned address with DRE disabled produces undefined results. DRE Support is only available for AXI4-Stream data width setting of 512-bits and under.

- **Use RxLength In Status Stream:** If the Control/Status Stream is enabled, checking this allows AXI DMA to use a receive length field that is supplied by the S2MM target IP in the App4 field of the status packet. This gives AXI DMA a pre-determined receive byte count, allowing AXI DMA to command the exact number of bytes to be transferred.

  This option provides for a higher bandwidth solution for systems needing greater throughput. In this configuration, the S2MM target IP can supply all data bytes specified in the receive length field of status packet APP4.

- **Enable Single AXI4 Data Interface:** This option is only applicable when used in the Vivado IP integrator. You can use this option to combine two AXI4 interfaces (MM2S and S2MM) into a single interface. This option does not affect the resource or performance.

User Parameters

The following table shows the relationship between the fields in the AMD Vivado™ IDE and the user parameters (which can be viewed in the Tcl Console).

Table 44: **Vivado IDE Parameter to User Parameter Relationship**

Vivado IDE Parameter/Value	User Parameter/Value	Default Value
Enable Scatter Gather Engine	c_include_sg	1
Enable Multi Channel Support	c_enable_multi_channel	0
Number of Channels (MM2S)	c_num_mm2s_channels	1
Number of Channels (S2MM)	c_num_s2mm_channels	1
Width of Buffer Length Register	c_sg_length_width	14
Enable Asynchronous clocks	c_prmry_is_aclk_async	0
Enable Control/Status Stream	c_sg_include_stscntrl_strm	1
Enable Micro DMA	c_micro_dma	0
Enable Read Channel	c_include_mm2s	1
Memory Map Data Width (MM2S)	c_m_axi_mm2s_data_width	32
Stream Data Width (MM2S)	c_m_axis_mm2s_tdata_width	32
Allow Unaligned Transfers (MM2S)	c_include_mm2s_dre	0
Max Burst Size (MM2S)	c_mm2s_burst_size	16
Enable Write Channel	c_include_s2mm	1

Table 44: **Vivado IDE Parameter to User Parameter Relationship** *(cont'd)*

Vivado IDE Parameter/Value	User Parameter/Value	Default Value
Use Rxlength in Status Stream	c_sg_use_stsapp_length	0
Memory Map Data Width (S2MM)	c_m_axi_s2mm_data_width	32
Stream Data Width (S2MM)	c_s_axis_s2mm_tdata_width	32
Allow Unaligned Transfers (S2MM)	c_include_s2mm_dre	0
Max Burst Size (S2MM)	c_s2mm_burst_size	16
Address Width (32-64)	c_addr_width	32

Output Generation

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896).

Constraining the Core

Necessary XDC constraints are delivered along with the core generation in the Vivado Design Suite.

Required Constraints

This section is not applicable for this IP core.

Device, Package, and Speed Grade Selections

This section is not applicable for this IP core.

Clock Frequencies

This section is not applicable for this IP core.

Clock Management

This section is not applicable for this IP core.

Clock Placement

This section is not applicable for this IP core.

Banking

This section is not applicable for this IP core.

Transceiver Placement

This section is not applicable for this IP core.

I/O Standard and Placement

This section is not applicable for this IP core.

Simulation

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

> **IMPORTANT!** *For cores targeting 7 series or Zynq 7000 devices, UNIFAST libraries are not supported. AMD IP is tested and qualified with UNISIM libraries only.*

Synthesis and Implementation

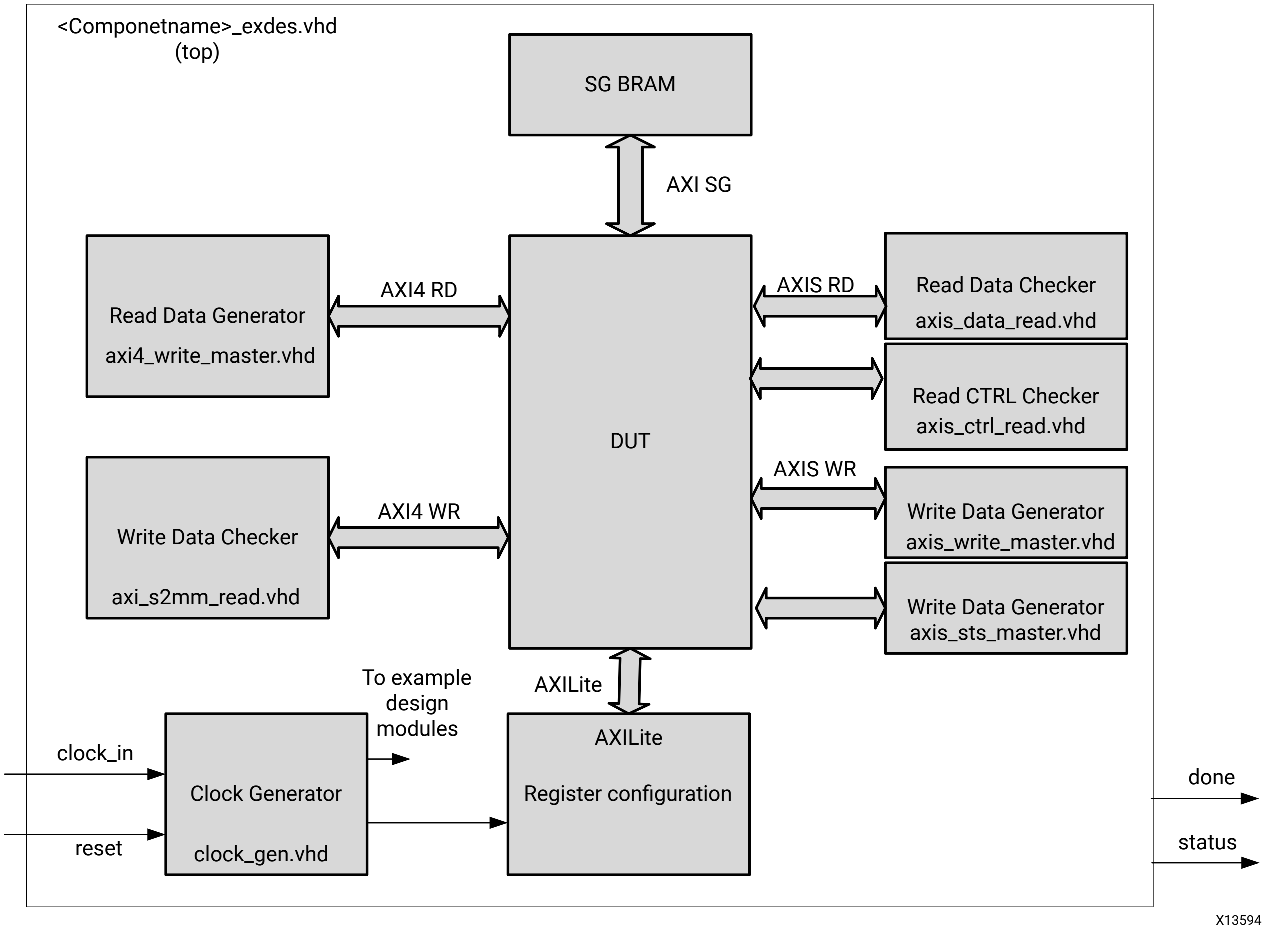
For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896).

Example Design

This chapter contains information about the example design provided in the AMD Vivado™ Design Suite.

The top module instantiates all components of the core and example design that are needed to implement the design in hardware, as shown in the following figure. This includes mixed-mode clock manager (MMCME2), register configuration, data generator, and data checker modules.

Figure 51: **Block Diagram of Example Design**



This example design demonstrates transactions on the AXI4-Lite, AXI4, and AXI4-Stream interfaces of the DUT.

- **Clock generator:** MMCME2 is used to generate the clocks for the example design. When the DUT is in synchronous mode, MMCME2 generates a 100 MHz clock for all the AXI interfaces in the example design. When in asynchronous mode, MMCME2 generates a 50 MHz clock for the AXI4-Lite interface and a 100 MHz clock for the AXI4 and AXI4-Stream interfaces.

  The DUT and other modules of the example design are kept under reset until MMCME2 is locked.

- **Register configuration module:** This module configures the DUT registers as mentioned in the programming sequence in Programming Sequence. This module is an AXI Traffic Generator module that is configured to program the registers. For more information, refer to the *AXI Traffic Generator LogiCORE IP Product Guide* (PG125).
- **Read path generator:** This uses an AXI block RAM which is filled (with a fixed amount of transfers) after MMCME2 is locked. MM2S channel reads this AXI block RAM and transfers data to the AXI4-Stream interface.
- **Read path checker:** This module checks the data transferred on the MM2S AXI4-Stream interface.
- **Read path CTRL checker:** This module checks the data transferred on the MM2S AXI4-Stream Control Interface.
- **Write path generator:** When the Write (S2MM) channel is configured, this module drives the transactions (with a fixed amount of transfers) on the S2MM AXI4-Stream interface.
- **Write path STS generator:** This module generates the S2MM STS data stream.
- **Write path checker:** This module checks the data received on the AXI4 interface. Data received on the AXI4 interface is also written into another AXI block RAM.

The test starts soon after the MMCME2 is locked. The Done pin is asserted High after all the transactions are completed. Similarly the Status pin is asserted High, when the data integrity check is successful. These two pins can be connected to LEDs to know the status of the test.

Implementing the Example Design

After following the steps described in Chapter 5: Design Flow Steps, implement the example design as follows:

1. Right-click the core in the Hierarchy window, and select Open IP Example Design.
2. A new window pops up, asking you to specify a directory for the example design. Select a new directory, or keep the default directory.

   A new project is automatically created in the selected directory and opened in a new Vivado IDE window.

3. In the Flow Navigator (left-side pane), click Run Implementation ,and follow the directions.

The following tables describe the files in the example design, simulation, and constraints directories.

In the current project directory, a new project with the name `<component_name>_example` is created and the files are delivered in that directory. This directory and its subdirectories contain all the source files that are required to create the AXI DMA controller example design.

The following table shows the files that are part of the example design.

Table 45: **Example Design Directory**

Name	Description
<component_name>_exdes.vhd	Top-level HDL file for the example design.
axi_lite_sm.vhd	Register configuration file for example design. (This file is not used by default. You can update the <component_name>_exdes.vhd to use in place of AXI Traffic Generator).
clock_gen.vhd	Clock generation module for example design.
axi4_write_master.vhd	Read path data generator module for example design.
axis_data_read.vhd	Read path data checker module for example design.
axis_write_master.vhd	Write path data generator module for example design.
axi_s2mm_read.vhd	Write path data checker module for example design.
axis_ctrl_read.vhd	MM2SCTRL data checker
axis_sts_master.vhd	S2MM STS data generator
sg_mif.coe	COE file used by block memory to store BDs

The following table shows the test bench file that can be used to run the simulation.

Table 46: **Simulation Directory**

Name	Description
<component_name>_exdes_tb.vhd	Test Bench for the example design

The following table shows the XDC file that is needed to implement the example design.

Table 47: **Constraints Directory**

Name	Description
<component_name>_exdes.xdc	Top level constraints file for the example design.

The XDC delivered with the example design has all the I/O pins configured for KC705 board. These constraints are commented by default. Uncomment them before proceeding with implementation for the KC705 board.

Simulating the Example Design

Using the AXI DMA example design (delivered as part of the AXI DMA), the behavior of the AXI DMA can be quickly simulated and observed.

The simulation script compiles the AXI DMA example design and supporting simulation files. It then runs the simulation and checks to ensure that it completed successfully.

If the test fails, the following message displays: `Test Failed!!!`

If the test passes, the following message displays: `Test Completed Successfully`

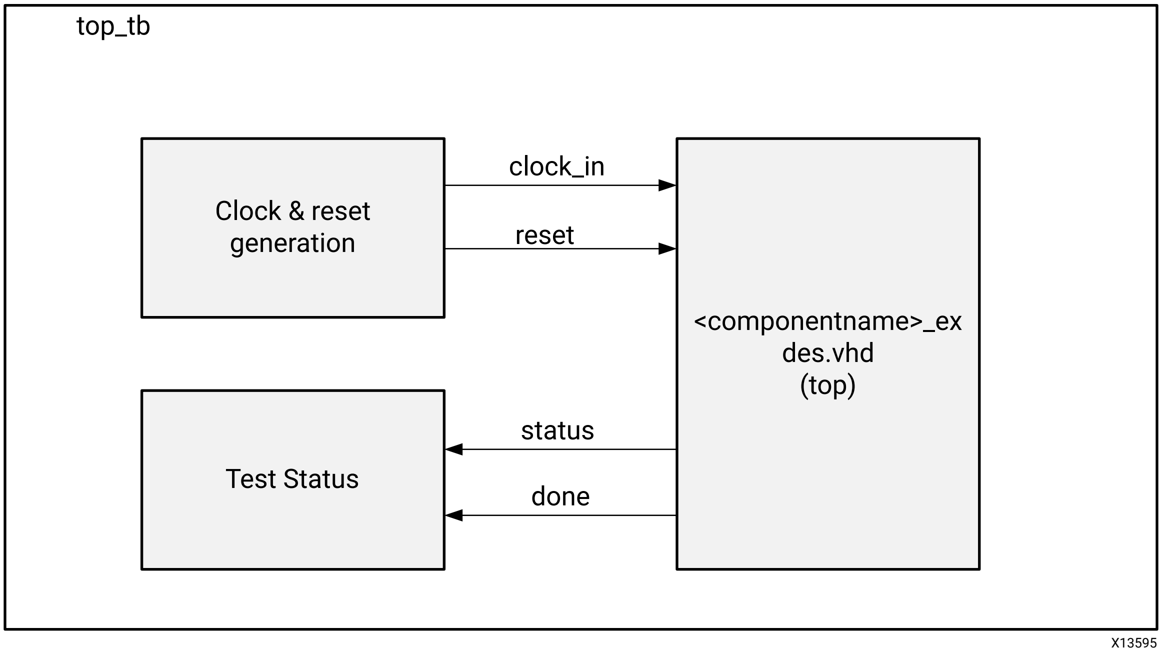
If the test hangs, the following message displays: `Test Failed!! Test Timed Out`

Test Bench for the Example Design

This section contains information about the provided test bench in the Vivado Design Suite.

The following figure shows the test bench for the AXI DMA example design. The top-level test bench generates a 200 MHz differential clock and drives an initial reset to the example design.

Figure 52: **AXI DMA Example Design Test Bench**



Appendix A

Upgrading

This appendix is not applicable for the first release of the core.

Appendix B

Debugging

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

If the IP requires a license key, the key must be verified. The AMD Vivado™ design tools have several license checkpoints for gating licensed IP through the flow. If the license check succeeds, the IP can continue generation. Otherwise, generation halts with an error. License checkpoints are enforced by the following tools:

- Vivado Synthesis
- Vivado Implementation
- write_bitstream (Tcl command)

IMPORTANT! *IP license level is ignored at checkpoints. The test confirms a valid license exists. It does not check IP license level.*

Finding Help with AMD Adaptive Computing Solutions

To help in the design and debug process when using the core, the Support web page contains key resources such as product documentation, release notes, answer records, information about known issues, and links for obtaining further product support. The Community Forums are also available where members can learn, participate, share, and ask questions about AMD Adaptive Computing solutions.

Documentation

This product guide is the main document associated with the core. This guide, along with documentation related to all products that aid in the design process, can be found on the AMD Adaptive Support web page or by using the AMD Adaptive Computing Documentation Navigator. Download the Documentation Navigator from the Downloads page. 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 an AMD Adaptive Computing product. Answer Records are created and maintained daily to ensure that users have access to the most accurate information available.

Answer Records for this core can be located by using the Search Support box on the main AMD Adaptive Support web page. To maximize your search results, use 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 Core

AR 54682.

Technical Support

AMD Adaptive Computing provides technical support on the Community Forums for this AMD LogiCORE™ IP product when used as described in the product documentation. AMD Adaptive Computing cannot guarantee timing, functionality, or support if you do any of the following:

- Implement the solution in devices that are not defined in the documentation.
- Customize the solution beyond that allowed in the product documentation.
- Change any section of the design labeled DO NOT MODIFY.

To ask questions, navigate to the Community Forums.

Debug Tools

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

Vivado Design Suite Debug Feature

The AMD Vivado™ Design Suite debug feature inserts logic analyzer and virtual I/O cores directly into your design. The debug feature also allows you to set trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be analyzed. This feature in the Vivado IDE is used for logic debugging and validation of a design running in AMD devices.

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)

See the *Vivado Design Suite User Guide: Programming and Debugging* (UG908).

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 AMD Vivado™ debug feature is a valuable resource to use in hardware debug. The signal names mentioned in the following individual sections can be probed using the debug feature for debugging the specific problems.

Some of the common problems encountered and possible solutions follow.

- You have programmed your BD ring but nothing seems to work. Register programming sequence has to be followed to start the DMA. See Programming Sequence and Descriptor Management.
- Internal Error/Error bits set in the Status register
    - Internal error will be set when BTT specified in the descriptor is 0.
    - SG internal error will be set if the fetched BD is a completed BD.
    - Other error bits like Decode Error or Slave Error would also be set based on the response from Interconnect or Slave.
- You are reading data from a location, but the data does not seem to be in order.

  Verify if the start address location is aligned or unaligned. If it is not aligned, ensure that the DRE is enabled while configuring DMA.

Also see the Solution Centers for support on devices, software tools, and intellectual property at all stages of the design cycle. Topics include design assistance, advisories, and troubleshooting tips.

Appendix C

Additional Resources and Legal Notices

Finding Additional Documentation

Technical Information Portal

The AMD Technical Information Portal is an online tool that provides robust search and navigation for documentation using your web browser. To access the Technical Information Portal, go to https://docs.amd.com.

Documentation Navigator

Documentation Navigator (DocNav) is an installed tool that provides access to AMD Adaptive Computing documents, videos, and support resources, which you can filter and search to find information. To open DocNav:

- From the AMD Vivado™ IDE, select **Help → Documentation and Tutorials**.
- On Windows, click the **Start** button and select **Xilinx Design Tools → DocNav**.
- At the Linux command prompt, enter `docnav`.

***Note*:** For more information on DocNav, refer to the *Documentation Navigator User Guide* (UG968).

Design Hubs

AMD Design Hubs provide links to documentation organized by design tasks and other topics, which you can use to learn key concepts and address frequently asked questions. To access the Design Hubs:

- In DocNav, click the **Design Hubs View** tab.
- Go to the Design Hubs web page.

Support Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Support.

References

These documents provide supplemental material useful with this guide:

1. *Vivado Design Suite User Guide: Designing with IP* (UG896)
2. *Vivado Design Suite: AXI Reference Guide* (UG1037)
3. *AMBA AXI and ACE Protocol Specification* (ARM IHI 0022D)
4. *Vivado Design Suite User Guide: Getting Started* (UG910)
5. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
6. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
7. *AXI Traffic Generator LogiCORE IP Product Guide* (PG125)
8. *ISE to Vivado Design Suite Migration Guide* (UG911)
9. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
10. *AXI Interconnect LogiCORE IP Product Guide* (PG059)
11. *AXI Multichannel Direct Memory Access LogiCORE IP Product Guide* (PG288)
12. *Vivado Design Suite User Guide: Release Notes, Installation, and Licensing* (UG973)

Revision History

The following table shows the revision history for this document.

Section	Revision Summary
06/24/2025 Version 7.1	
Typical System Interconnect	Editorial update.
06/20/2024 Version 7.1	
MM2S_CONTROL (MM2S Control)	Updated Burst Length field equation.
MM2S_STATUS (MM2S Status)	Updated Transferred Bytes field description.
S2MM_CONTROL (S2MM Control)	Updated Buffer Length field description.
Field Descriptions	Updated Width of Buffer Length Register description.

Section	Revision Summary
04/27/2022 Version 7.1	
Performance	Updated section.
Scatter Gather Descriptor	Updated section.
06/14/2019 Version 7.1	
Entire document	Updated figures.
MM2S_DMASR (MM2S DMA Status Register – Offset 04h)	Updated table.
MM2S_DMASR (MM2S DMA Status Register – Offset 04h)	Updated table.
Field Descriptions	Added Enable Single AXI4 Data Interface section.
04/04/2018 Version 7.1	
N/A	Added support for 64 MB data transfer.
10/04/2017 Version 7.1	
N/A	• Added Documentation Navigator and Design Hubs to this appendix. • Added Automotive Applications Disclaimer. • Updated Data Re-Alignment Engine support to 512 bits (was 64 bits).
10/05/2016 Version 7.1	
N/A	• Added a note about the AXI4-Lite write access register to the beginning of the Register Space section. • Updated S2MM description.
11/18/2015 Version 7.1	
N/A	Added support for UltraScale+ families.
04/01/2015 Version 7.1	
N/A	• Fixed link to master answer record. • Added support for 64-bit addressing.
04/02/2014 Version 7.1	
N/A	• Added information about the Air Traffic Generator. • Added information about optional Micro DMA. • Added axi_dma_tstvec to I/O signals.
12/18/2013 Version 7.1	
N/A	Added AMD UltraScale™ architecture support.
10/02/2013 Version 7.1	
N/A	• Added example design • Added Cyclic BD Enable. • Modified Bits 26 and 27 of the S2MM_CONTROL register. • Updated screen displays. • Added IP integrator information. • Added Enable Micro DMA option.

Section	Revision Summary
03/20/2013 Version 7.0	
N/A	- Revision number advanced to 7.0 to align with core version number 7.0. - Updated for Vivado Design Suite support and core version 7.0 - Updated Debugging appendix. - Removed one screen capture and updated another in Chapter 4. - Removed ISE™, CORE Generator™, Virtex™-6, and AMD Spartan™- 6 material. - Removed Design Parameters and AXI DMA System Configuration sections from Chapter 3.
12/18/2012 Version 3.2	
N/A	- Updated for 14.4/2012.4 support and core version 6.03a. - Updated Debugging appendix. - Updated screen captures in Chapter 4. - Replaced Figure 1-1. - Updated devices in Table 2-1, System Performance. - Updated resource numbers in Tables 2-4, 2-5, and 2-6. - Removed Interconnect Parameters and Allowable Parameter Combinations sections. - Updated Output Generation sections in Chapters 4 and 7
10/16/2012 Version 3.1	
N/A	- Updated for 14.3/2012.3 support. - Document cleanup
07/25/2012 Version 3.0	
N/A	Added Vivado tools support and AMD Zynq™ 7000 support.
04/24/2012 Version 2.0	
N/A	- Added multichannel support - Added 2-D transactions support - Added keyhole support - Added Cache and User controls for AXI memory side Interface
10/19/2011 Version 1.0	
Initial release.	N/A

Please Read: Important Legal Notices

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions, and typographical errors. The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. Any computer system has risks of security vulnerabilities that cannot be completely prevented or mitigated. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes. THIS INFORMATION IS PROVIDED "AS IS." AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS, OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION. AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY RELIANCE, DIRECT, INDIRECT, SPECIAL, OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

AUTOMOTIVE APPLICATIONS DISCLAIMER

AUTOMOTIVE PRODUCTS (IDENTIFIED AS "XA" IN THE PART NUMBER) ARE NOT WARRANTED FOR USE IN THE DEPLOYMENT OF AIRBAGS OR FOR USE IN APPLICATIONS THAT AFFECT CONTROL OF A VEHICLE ("SAFETY APPLICATION") UNLESS THERE IS A SAFETY CONCEPT OR REDUNDANCY FEATURE CONSISTENT WITH THE ISO 26262 AUTOMOTIVE SAFETY STANDARD ("SAFETY DESIGN"). CUSTOMER SHALL, PRIOR TO USING OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE PRODUCTS, THOROUGHLY TEST SUCH SYSTEMS FOR SAFETY PURPOSES. USE OF PRODUCTS IN A SAFETY APPLICATION WITHOUT A SAFETY DESIGN IS FULLY AT THE RISK OF CUSTOMER, SUBJECT ONLY TO APPLICABLE LAWS AND REGULATIONS GOVERNING LIMITATIONS ON PRODUCT LIABILITY.

Copyright