LogiCORE IP AXI DMA v5.00.a

Product Guide

PG021 October 19, 2011

XILINX®

Table of Contents

Chapter 1: Overview

Chapter 2: Core Interfaces and Register Space

Chapter 3: Customizing and Generating the Core

Chapter 4: Designing with the Core

Chapter 5: Constraining the Core

Chapter 6: Detailed Example Design

Appendix 7: Additional Resources

Introduction

The Advanced eXtensible Interface (AXI) Direct Memory Access (AXI DMA) core is a soft Xilinx Intellectual Property (IP) core for use with the Xilinx® Embedded Development Kit (EDK) and the CORE Generator™ tools. The AXI DMA engine provides high-bandwidth direct memory access between memory and AXI4-Stream-type target peripherals. Its optional scatter gather capabilities also off-load data movement tasks from the Central Processing Unit (CPU). Initialization, status, and management registers are accessed through an AXI4-Lite slave interface, suitable for the Xilinx MicroBlaze™ microprocessor.

Features

- AXI4 compliant
- Optional Independent Scatter/Gather Direct Memory Access (DMA) support
- Optional Simple DMA Support
- Primary AXI4 Memory Map data width support of 32, 64, 128, 256, 512 and 1024 bits
- Primary AXI4-Stream data width support of 8, 16, 32, 64, 128, 256, 512 and 1024 bits
- Optional Data Re-Alignment Engine
- Optional AXI Control and Status Streams
- Use with Xilinx Platform Studio and Xilinx CORE Generator tool

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family[(1)(2)]	Virtex®-7, Kintex™-7, Virtex-6, Spartan®-6
Supported User Interfaces	AXI4, AXI4-Lite, AXI4-Stream
Resources	**See Table 1-4 and Table 1-5**
Provided with Core	
Design Files	VHSIC Hardware Description Language (VHDL)
Example Design	Not Provided
Test Bench	Not Provided
Constraints File	Not Provided
Simulation Model	Not Provided
Tested Design Tools[(3)]	
Design Entry Tools	EDK Xilinx Platform Studio (XPS) Integrated Software Environment (ISE®)
Simulation	QuestSim-64
Synthesis Tools	Xilinx Synthesis Technology (XST)
Support	
Provided by Xilinx, Inc.	

1. For a complete list of supported EDK derivative devices, see IDS Embedded Edition Derivative Device Support.
2. For a complete listing of supported devices for IP cores, see the release notes for this core.
3. For the supported versions of the tools, see the ISE Design Suite 13: Release Notes Guide.

Overview

'The AXI Direct Memory Access (AXI DMA) IP provides high-bandwidth direct memory access between the AXI4 memory mapped and AXI4-Stream IP interfaces. Its optional scatter gather capabilities also off load data movement tasks from the Central Processing Unit (CPU) in processor based systems. Initialization, status, and management registers are accessed through an AXI4-Lite slave interface. Figure 1-1 illustrates the functional composition of the core. The core's design has four AXI4 Memory Map interfaces:

- AXI4-Lite Slave
- AXI4 Memory Map Read Master
- AXI4 Memory Map Write Master
- Optional AXI4 Memory Map Scatter/Gather Read/Write Master

Associated with the memory map interfaces are four AXI4-Stream interfaces:

- AXI4 Memory Map to Stream (MM2S) Stream Master
- AXI4-Stream to Memory Map (S2MM), Stream Slave
- AXI Control Stream Master
- AXI Status Stream Slave

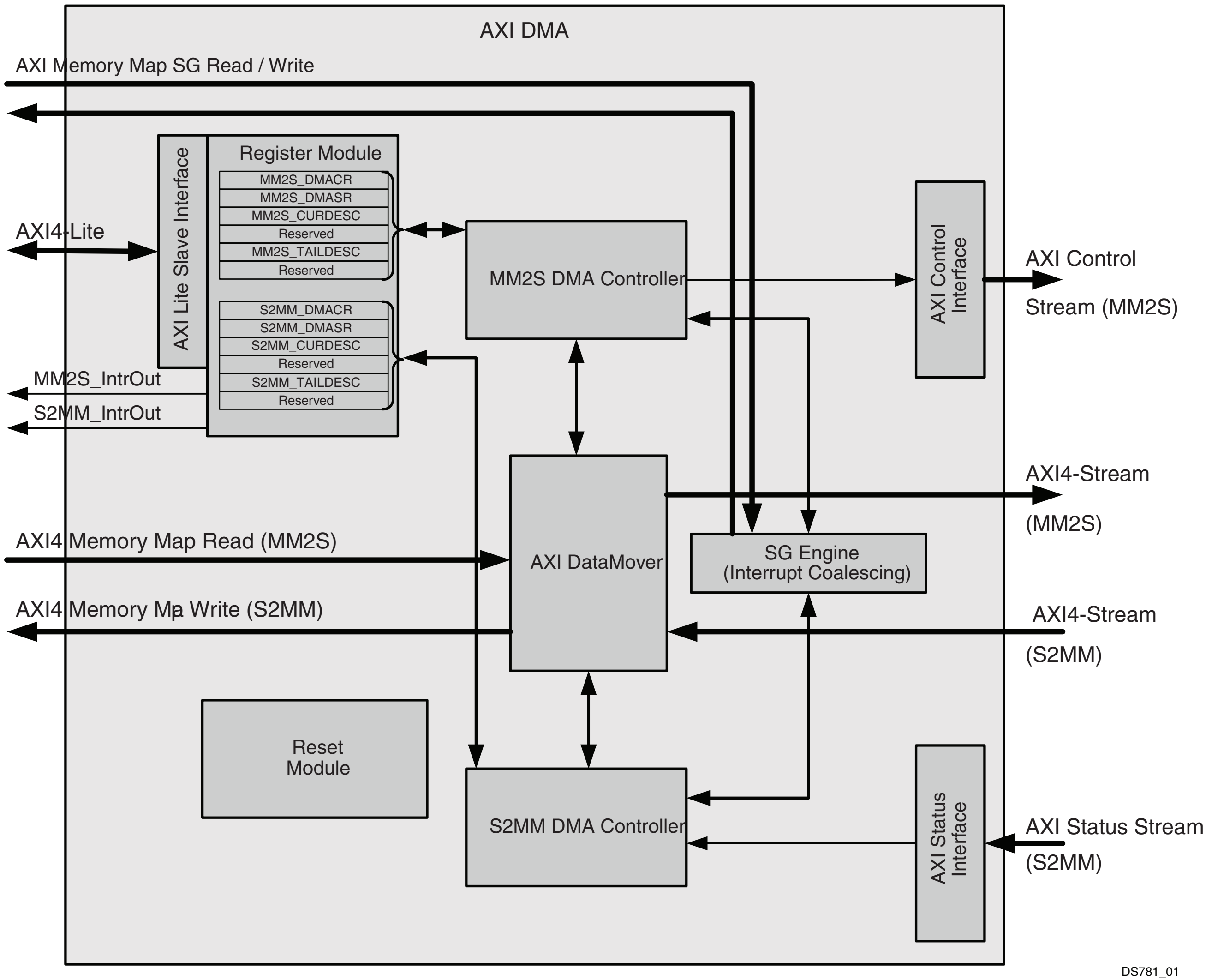


Figure 1-1: **AXI DMA Block Diagram**

Primary high-speed DMA data movement between system memory and stream target is through the AXI4 Memory Map Read Master to AXI MM2S Stream Master, and AXI S2MM Stream Slave to AXI4 Memory Map Write Master. The AXI DataMover is used for high throughput transfer of data from memory to stream and from stream to memory. The MM2S channel and S2MM channel operate independently and in a full-duplex-like method. The AXI DataMover provides the AXI DMA with 4 kbyte address boundary protection, automatic burst partitioning, 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 DataMover provides byte-level data realignment allowing memory reads and writes to any byte offset location.

Associated with each primary data channel is a stream channel for off-loading packet metadata from the primary datapath. 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 Memory Map Scatter Gather Read/Write Master interface. Optional descriptor queuing is provided to maximize primary data throughput.

Typical System Interconnect

The AXI DMA core is designed to be connected via the AXI Interconnect in the user's system. A typical MicroBlaze™ processor configuration is shown in Figure 1-2. The system's microprocessor has access to the AXI DMA through the AXI4-Lite interface. An integral Scatter/Gather Engine fetches buffer descriptors from DDRx which then coordinates primary data transfers between AXI IP and DDRx. Separate control and status streams provide packet-associated information, such as checksum off-load control/status, to and from AXI IP. The dual interrupt output of the AXI DMA core is routed to the System Interrupt Controller.

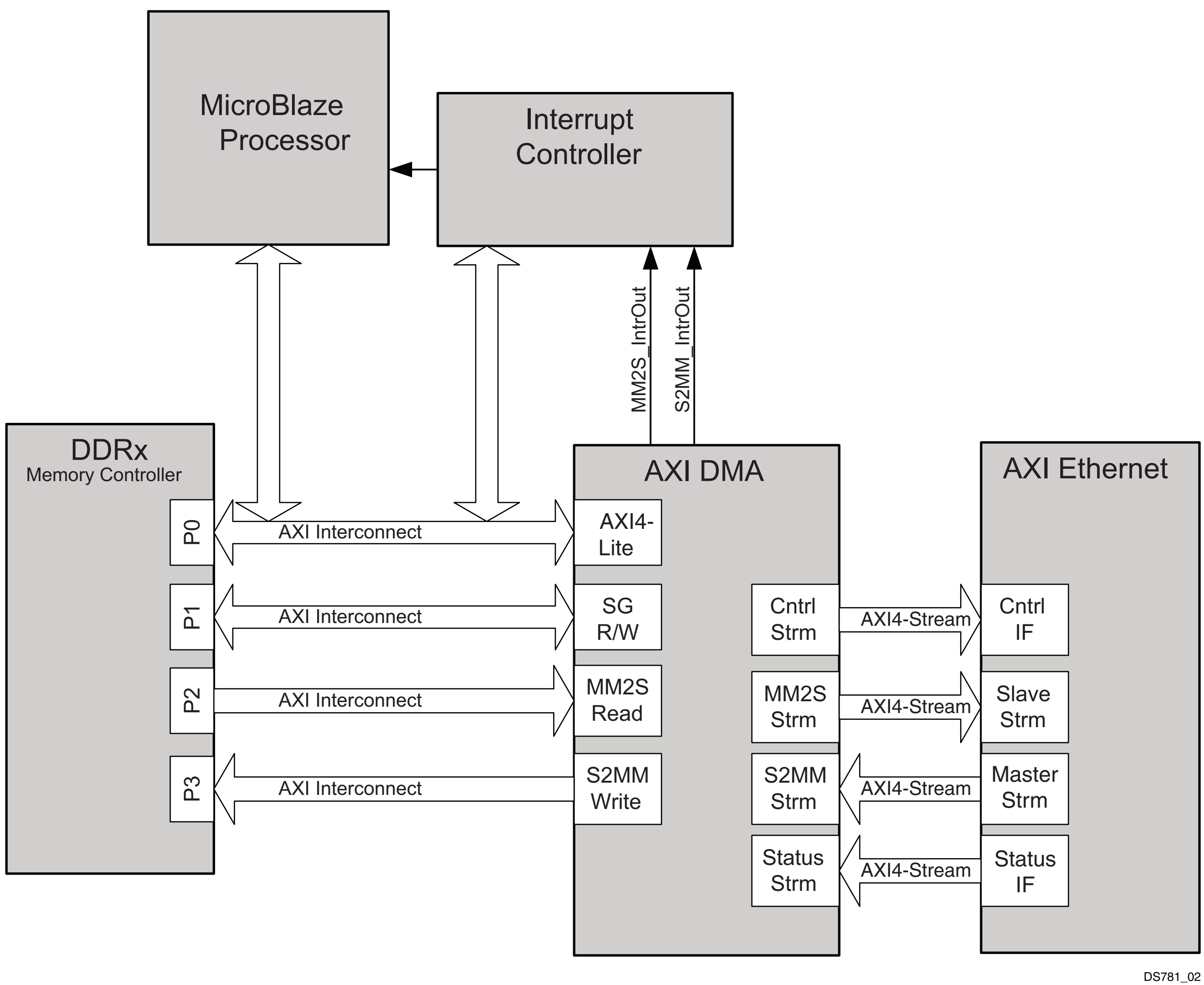


Figure 1-2: **Typical MicroBlaze Processor System Configuration (AXI Ethernet)**

The AXI DMA core can also be connected to a user's system other than with an Ethernet-based AXI IP. In this case, the parameter C_SG_INCLUDE_STSCNTRL_STRM should be set to 0 to exclude status and control information and use it for data only. A typical MicroBlaze processor configuration with general purpose AXI IP is shown in Figure 1-3.

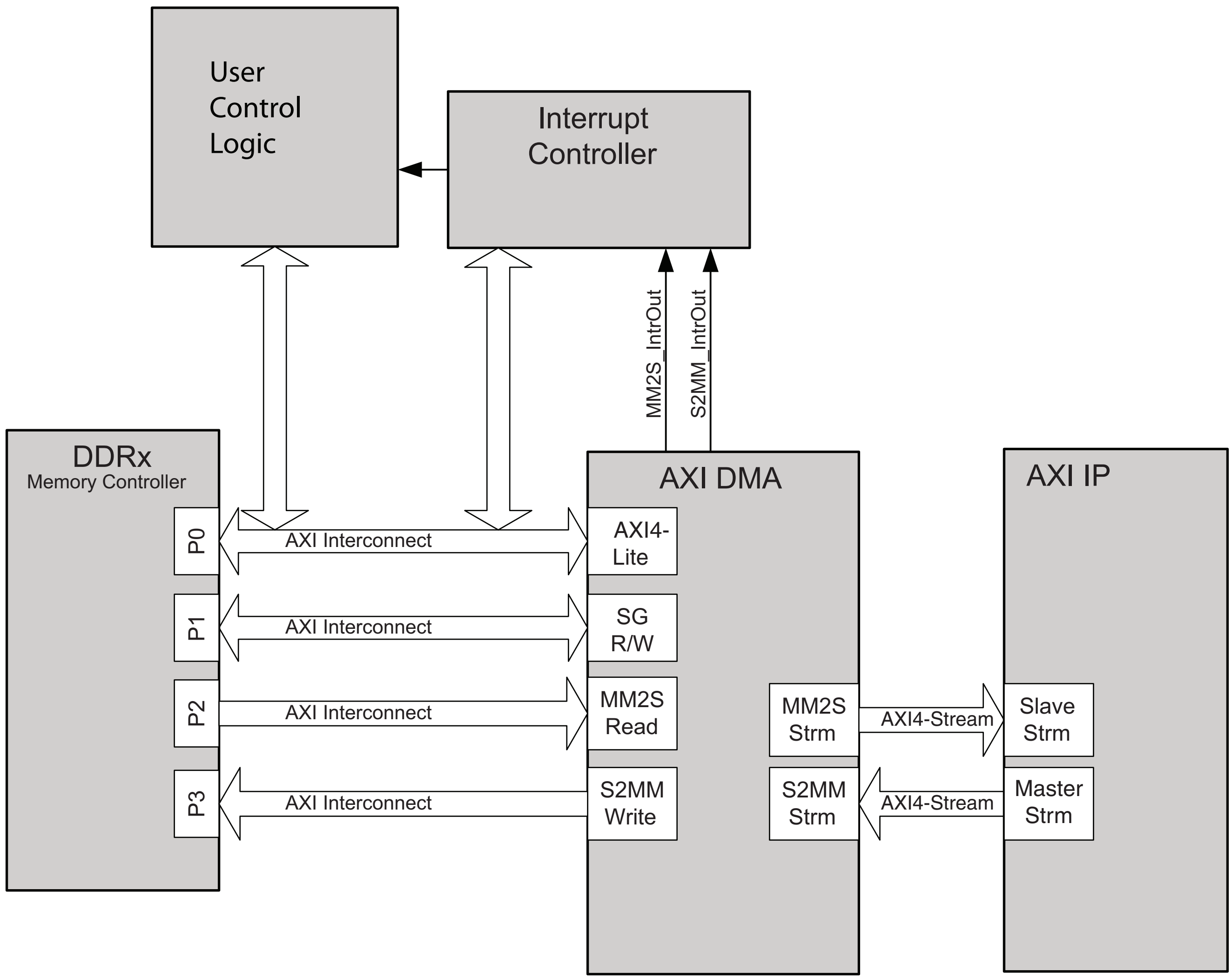


Figure 1-3: **Typical MicroBlaze Processor System Configuration (AXI IP)**

Operating System Requirements

For a list of System Requirements, see ISE Design Suite 13: Release Notes Guide.

Feature Summary

- AXI4 compliant
- Optional Independent Scatter/Gather Direct Memory Access (DMA) support
    - Provides off-loading 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 local transfer descriptor queuing
    - Provides optional User Application Fields
    - Provides Tail Descriptor Mode
- Optional Simple DMA Support

  A lower performance but less FPGA resource intensive mode can be enabled by excluding the Scatter Gather engine (C_INCLUDE_SG = 0). This mode is referenced in this document as Simple DMA mode. 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.

  A non-zero write to the length register for the specific channel triggers the transfer. On MM2S, length bytes are read from the Source Address and transmitted out the MM2S AXI Stream Interface.

  On S2MM, the length value indicates to AXI DMA the size of the storage buffer in bytes; therefore, length values that are written must be equal to or larger than the largest packet that is received on S2MM. A length value written that is smaller than the received packet produces undefined results.

  On S2MM the number of bytes in the received packet are written to the Destination Address and the length register is updated with the number of bytes transferred.
- Primary AXI4 Memory Map data width support of 32, 64, 128, 256, 512 and 1024 bits
- Primary AXI4-Stream data width support of 8, 16, 32, 64, 128, 256, 512 and 1024 bits
- Optional Data Re-Alignment Engine
    - Allows data realignment to the byte (8 bits) level on the primary memory map and stream datapaths
- Optional AXI Control and Status Streams
    - Provides optional Control Stream for MM2S Channel and Status Stream for the S2MM channel to off-load low-bandwidth control and status from high-bandwidth datapath.

Applications

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

Standards Compliance

AXI4, AXI4-Lite and AXI4-Stream compliant

Licensing

No license is required.

Performance

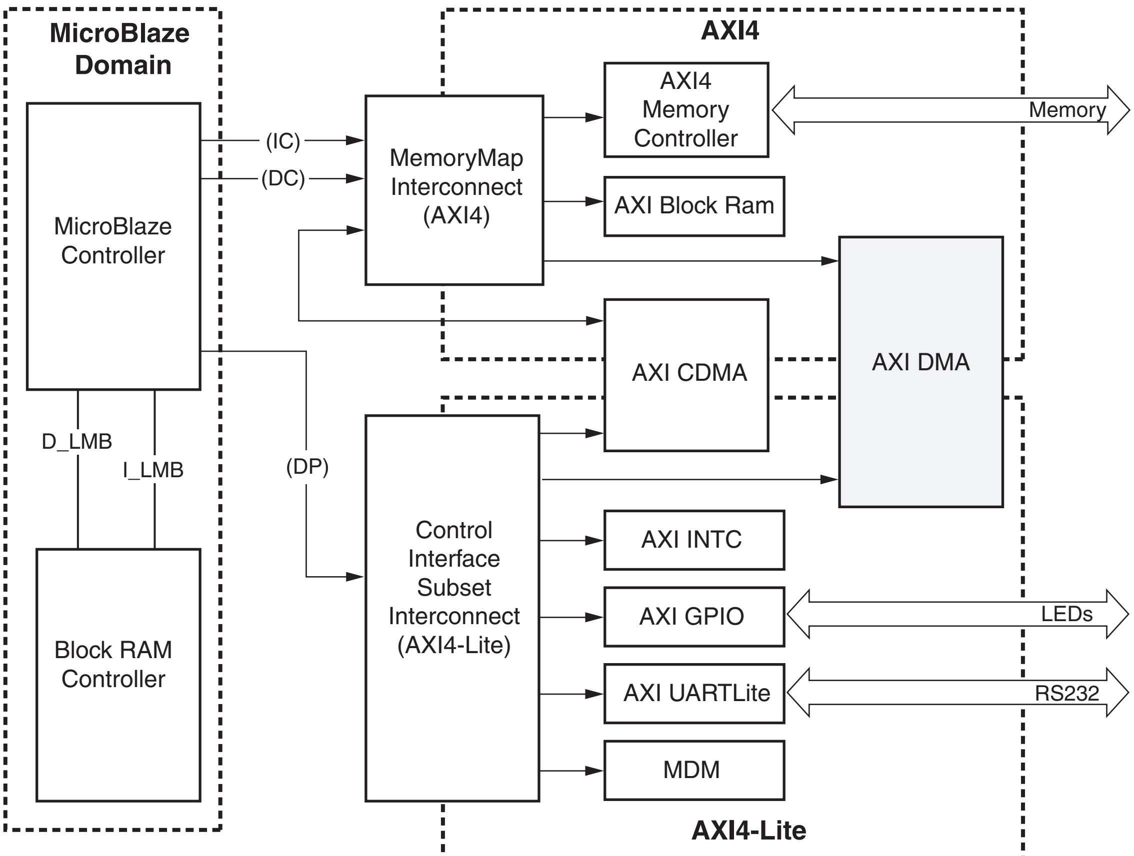


Figure 1-4: **Virtex-6 and Spartan-6 FPGA System Configuration Diagram**

Maximum Frequencies

The target FPGA was filled with logic to drive the Lookup Table (LUT) and block RAM utilization to approximately 70% and the I/O utilization to approximately 80%. Using the default tool options and the slowest speed grade for the target FPGA, the resulting target F_{MAX} numbers are shown in Table 1-1.

Table 1-1: **System Performance**

Target FPGA	Target F_{MAX} (MHz)		
	AXI4	AXI4-Lite	MicroBlaze
xc6slx45t [(1)]	90	120	80
xc6vlx240t [(2)]	135	180	135

Notes:

1. Spartan-6 FPGA LUT utilization: 70%; Block RAM utilization: 70%; I/O utilization: 80%; MicroBlaze processor not AXI4 interconnect; AXI4 interconnect configured with a single clock of 120 MHz.
2. Virtex-6 FPPGA LUT utilization: 70%; Block RAM utilization: 70%; I/O utilization: 80%.

Latency and Throughput

Table 1-2 describes the latency and throughput for the AXI DMA. The table provides performance information for typical and high performance configurations. Throughput tests consisted of 256 descriptors for each channel with each descriptor describing a 9000 byte packet.

Typical Configuration

- C_MM2S_DATA_WIDTH = 32 and C_S2MM_DATA_WIDTH = 32
- C_MM2S_BURST_SIZE = 32 and C_S2MM_BURST_SIZE = 32
- C_SG_INCLUDE_DESC_QUEUE = 1
- C_SG_INCLUDE_STSCNTRL_STRM = 1
- C_SG_USE_STSAPP_LENGTH = 1
- C_INCLUDE_SG = 1

High Performance Configuration

- C_MM2S_DATA_WIDTH = 32 and C_S2MM_DATA_WIDTH = 32
- C_MM2S_BURST_SIZE = 128 and C_S2MM_BURST_SIZE = 128
- C_SG_INCLUDE_DESC_QUEUE = 1
- C_SG_INCLUDE_STSCNTRL_STRM = 1
- C_SG_USE_STSAPP_LENGTH = 1
- C_INCLUDE_SG = 1

Table 1-2: **AXI DMA Latency and Throughput**

AXI DMA Configuration	Primary Clock Frequency (axi_prmry_aclk)	Packet Size (Bytes)	Initial Transaction Latency (axi_prmry_aclk Clocks)[1]	Maximum Total Data Throughput (MBytes/sec)
Spartan-6 FPGAs				
Typical Configuration	100 MHz	9000	9	274.8
High Performance Configuration	100 MHz	9000	9	344.4
Virtex-6 FPGAs				
Typical Configuration	150 MHz	9000	9	301.1
High Performance Configuration	150 MHz	9000	9	467.5

Notes:

1. Latency measured via simulation from TAILDESC register update to m_axi_sg_arvalid assertion.

Resource Utilization

Resources required for the AXI DMA core have been estimated for the Virtex®-7 (Table 1-3), Virtex-6 (Table 1-4) and Spartan®-6 FPGAs (Table 1-5). These values were generated using the Xilinx® EDK tools v13.3. They are derived from post-MAP reports and can change during PAR.

Table 1-3: **Virtex-7 FPGA Resource Estimates**

C_M_AXI_MM2S_TDATA_WIDTH	C_S_AXI_S2MM_TDATA_WIDTH	C_M_AXI_MM2S_DATA_WIDTH	C_M_AXI_S2MM_DATA_WIDTH	C_MM2S_BURST_SIZE	C_S2MM_BURST_SIZE	C_SG_INCLUDE_DESC_QUEUE	C_INCLUDE_SGSCNTRL_STRM	C_SG_USE_STSAPP_LENGTH	C_INCLUDE_MM2S_DRE	C_INCLUDE_S2MM_DRE	C_INCLUDE_SG	C_PRMRY_IS_ACLK_ASYNC	Slices	Slice Reg	Slice LUTs	Block RAM
32	32	32	32	16	16	0	1	1	0	0	1	1	1693	4516	3270	3
16	16	32	32	16	16	1	1	1	0	0	1	1	1868	4855	3644	3
16	16	32	32	16	16	1	1	1	1	1	1	1	1988	4973	3709	3
32	32	64	64	16	16	1	1	1	0	0	1	1	1996	5131	3803	4
32	32	32	32	16	16	0	0	0	0	0	0	1	1003	2468	1741	2

Table 1-4: **Virtex-6 FPGA Resource Estimates**

C_M_AXI_MM2S_TDATA_WIDTH	C_S_AXI_S2MM_TDATA_WIDTH	C_M_AXI_MM2S_DATA_WIDTH	C_M_AXI_S2MM_DATA_WIDTH	C_MM2S_BURST_SIZE	C_S2MM_BURST_SIZE	C_SG_INCLUDE_DESC_QUEUE	C_INCLUDE_SGSCNTRL_STRM	C_SG_USE_STSAPP_LENGTH	C_INCLUDE_MM2S_DRE	C_INCLUDE_S2MM_DRE	C_INCLUDE_SG	C_PRMRY_IS_ACLK_ASYNC	Slices	Slice Reg	Slice LUTs	Block RAM
32	32	32	32	16	16	0	1	1	0	0	1	1	1834	4516	3029	3
16	16	32	32	16	16	1	1	1	0	0	1	1	2033	4860	3450	3
16	16	32	32	16	16	1	1	1	1	1	1	1	2130	4976	3518	3
32	32	64	64	16	16	1	1	1	0	0	1	1	2099	5134	3657	4
32	32	32	32	16	16	0	0	0	0	0	0	1	1151	2468	1641	2

Table 1-5: **Spartan-6 FPGA Resource Estimates**

C_M_AXI_MM2S_DATA_WIDTH	C_M_AXI_S2MM_DATA_WIDTH	C_MM2S_BURST_SIZE	C_S2MM_BURST_SIZE	C_SG_INCLUDE_DESC_QUEUE	C_INCLUDE_SGSCNTRL_STRM	C_SG_USE_STSAPP_LENGTH	C_INCLUDE_MM2S_DRE	C_INCLUDE_S2MM_DRE	C_INCLUDE_SG	C_PRMRY_IS_ACLK_ASYNC	Slices	Slice Reg	Slice LUTs	Block RAM
32	32	16	16	0	0	0	0	0	0	0	771	1743	1171	2
32	32	16	16	0	0	0	1	1	0	0	974	1985	1470	2
64	64	16	16	0	0	0	1	1	0	0	1393	2601	2053	3
128	128	16	16	0	0	0	0	0	0	0	1155	2627	1916	5
256	256	16	16	0	0	0	0	0	0	0	1563	3674	2675	8
32	32	16	16	0	0	0	0	0	1	0	1468	3368	2186	2
32	32	16	16	1	0	0	0	0	1	0	1655	3736	2637	4
32	32	16	16	1	1	0	0	0	1	0	1794	4111	2927	5
32	32	16	16	1	1	1	0	0	1	0	1678	3933	2775	3
32	32	16	16	1	1	1	1	1	1	0	1881	4187	3149	3
64	64	16	16	1	1	1	1	1	1	0	2247	4738	3693	3
128	128	16	16	1	1	1	0	0	1	0	1859	4555	3098	3
256	256	16	16	1	1	1	0	0	1	0	2125	5599	3689	3
256	256	32	32	1	1	1	0	0	1	0	2143	5607	3705	3
256	256	64	64	1	1	1	0	0	1	0	2151	5615	3717	3
256	256	128	128	1	1	1	0	0	1	0	2127	5623	3698	3
256	256	256	256	1	1	1	0	0	1	0	2101	5623	3698	3
256	256	256	256	1	1	1	0	0	1	1	2443	6951	3888	9

Chapter 2

Core Interfaces and Register Space

This chapter provides detailed descriptions for each interface. In addition, detailed information about configuration and control registers is included.

Port Descriptions

The AXI DMA I/O signals are described in Table 2-1.

Table 2-1: **I/O Signal Description**

Signal Name	Interface	Signal Type	Init Status	Description
s_axi_lite_aclk	Clock	I		AXI DMA AXI4-Lite Clock. Must be less than or equal to axi_sg_aclk for asynchronous mode. (C_PRMRYIS_ACLK_ASYNC=1).
m_axi_sg_aclk	Clock	I		AXI DMA Scatter Gather Clock. Scatter Gather clock must be less than or equal to the slowest of m_axi_mm2s_aclk or m_axi_s2mm_aclk for asynchronous mode. (C_PRMRY_IS_ACLK_ASYNC=1).
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.
AXI4-Lite Interface Signals				
s_axi_lite_awvalid	S_AXI_LITE	I		AXI4-Lite Write Address Channel Write Address Valid. • 1 = Write address is valid. • 0 = Write address is not valid.
s_axi_lite_awready	S_AXI_LITE	O	0	AXI4-Lite Write Address Channel Write Address Ready. Indicates DMA ready to accept the write address. • 1 = Ready to accept address. • 0 = Not ready to accept address.

Table 2-1: **I/O Signal Description *(Cont'd)***

Signal Name	Interface	Signal Type	Init Status	Description
s_axi_lite_awaddr(31:0)	S_AXI_LITE	I		AXI4-Lite Write Address Bus
s_axi_lite_wvalid	S_AXI_LITE	I		AXI4-Lite Write Data Channel Write Data Valid. • 1 = Write data is valid. • 0 = Write data is not valid.
s_axi_lite_wready	S_AXI_LITE	O	0	AXI4-Lite Write Data Channel Write Data Ready. Indicates DMA ready to accept the write data. • 1 = Ready to accept data. • 0 = Not ready to accept data.
s_axi_lite_wdata(31:0)	S_AXI_LITE	I		AXI4-Lite Write Data Bus
s_axi_lite_bresp(1:0)	S_AXI_LITE	O	zeros	AXI4-Lite Write Response Channel. Indicates results of the write transfer. The AXI DMA Lite interface always responds with OKAY. • 00b = OKAY - Normal access has been successful. • 01b = EXOKAY - Not supported. • 10b = SLVERR - Not supported. • 11b = DECERR - Not supported.
s_axi_lite_bvalid	S_AXI_LITE	O	0	AXI4-Lite Write Response Channel Response Valid. Indicates response is valid. • 1 = Response is valid. • 0 = Response is not valid.
s_axi_lite_bready	S_AXI_LITE	I		AXI4-Lite Write Response Channel Ready. Indicates target is ready to receive response. • 1 = Ready to receive response. • 0 = Not ready to receive response.
s_axi_lite_arvalid	S_AXI_LITE	I		AXI4-Lite Read Address Channel Read Address Valid. • 1 = Read address is valid. • 0 = Read address is not valid.
s_axi_lite_arready	S_AXI_LITE	O	0	AXI4-Lite Read Address Channel Read Address Ready. Indicates DMA ready to accept the read address. • 1 = Ready to accept address. • 0 = Not ready to accept address.
s_axi_lite_araddr(31:0)	S_AXI_LITE	I		AXI4-Lite Read Address Bus.
s_axi_lite_rvalid	S_AXI_LITE	O	0	AXI4-Lite Read Data Channel Read Data Valid 1 = Read data is valid 0 = Read data is not valid
s_axi_lite_rready	S_AXI_LITE	I		AXI4-Lite Read Data Channel Read Data Ready. Indicates target ready to accept the read data. • 1 = Ready to accept data. • 0 = Not ready to accept data.

Table 2-1: **I/O Signal Description *(Cont'd)***

Signal Name	Interface	Signal Type	Init Status	Description
s_axi_lite_rdata(31:0)	S_AXI_LITE	O	zeros	AXI4-Lite Read Data Bus
s_axi_lite_rresp(1:0)	S_AXI_LITE	O	zeros	AXI4-Lite Read Response Channel Response. Indicates results of the read transfer. The AXI DMA Lite interface always responds with OKAY. • 00b = OKAY - Normal access has been successful. • 01b = EXOKAY - Not supported • 10b = SLVERR - Not supported • 11b = DECERR - Not supported
MM2S Memory Map Read Interface Signals				
m_axi_mm2s_araddr (C_M_AXI_MM2S_ADDR_ WIDTH-1: 0)	M_AXI_MM2S	O	zeros	Read Address Channel Address Bus
m_axi_mm2s_arlen(7:0)	M_AXI_MM2S	O	zeros	Read Address Channel Burst Length. In data beats - 1.
m_axi_mm2s_arsize(2:0)	M_AXI_MM2S	O	zeros	Read Address Channel Burst Size. Indicates width of burst transfer. • 000b = 1 byte (8-bit wide burst) • 001b = 2 bytes (16-bit wide burst) • 010b = 4 bytes (32-bit wide burst) • 011b = 8 bytes (64-bit wide burst) • 100b = 16 bytes (128-bit wide burst) • 101b = 32 bytes (256-bit wide burst) • 110b = Not Supported by AXI DMA • 111b = Not Supported by AXI DMA
m_axi_mm2s_arburst(1:0)	M_AXI_MM2S	O	zeros	Read Address Channel Burst Type. Indicates type burst. • 00b = FIXED - Not supported • 01b = INCR - Incrementing address • 10b = WRAP - Not supported • 11b = Reserved
m_axi_mm2s_arprot(2:0)	M_AXI_MM2S	O	010b	Read Address Channel Protection. Always driven with a constant output of 010b.
m_axi_mm2s_arcache(3:0)	M_AXI_MM2S	O	0011b	Read Address Channel Cache. This is always driven with a constant output of 0011b.
m_axi_mm2s_arvalid	M_AXI_MM2S	O	0	Read Address Channel Read Address Valid. Indicates m_axi_mm2s_araddr is valid. • 1 = Read address is valid. • 0 = Read address is not valid.

Table 2-1: **I/O Signal Description *(Cont'd)***

Signal Name	Interface	Signal Type	Init Status	Description
m_axi_mm2s_arready	M_AXI_MM2S	I		Read Address Channel Read Address Ready. Indicates target is ready to accept the read address. • 1 = Target ready to accept address. • 0 = Target not ready to accept address.
m_axi_mm2s_rdata (C_M_AXI_MM2S_DATA_WIDTH-1: 0)	M_AXI_MM2S	I		Read Data Channel Read Data
m_axi_mm2s_rresp(1:0)	M_AXI_MM2S	I		Read Data Channel Response. Indicates results of the read transfer. • 00b = OKAY - Normal access has been successful. • 01b = EXOKAY - Not supported. • 10b = SLVERR - Slave returned error on transfer. • 11b = DECERR - Decode error, transfer targeted unmapped address.
m_axi_mm2s_rlast	M_AXI_MM2S	I		Read Data Channel Last. Indicates the last data beat of a burst transfer. • 1 = Last data beat • 0 = Not last data beat
m_axi_mm2s_rvalid	M_AXI_MM2S	I		Read Data Channel Data Valid. Indicates m_axi_mm2s_rdata is valid. • 1 = Valid read data • 0 = Not valid read data
m_axi_mm2s_rready	M_AXI_MM2S	O	0	Read Data Channel Ready. Indicates the read channel is ready to accept read data. • 1 = Ready • 0 = Not ready

Table 2-1: **I/O Signal Description *(Cont'd)***

Signal Name	Interface	Signal Type	Init Status	Description
MM2S Master Stream Interface Signals				
mm2s_prmry_reset_out_n	M_AXIS_MM2S	O	0	Primary MM2S Reset Out
m_axis_mm2s_tdata (C_M_AXIS_MM2S_TDATA_WIDTH-1: 0)	M_AXIS_MM2S	O	zeros	AXI4-Stream Stream Data Out
m_axis_mm2s_tkeep (C_M_AXIS_MM2S_TDATA_WIDTH/8-1: 0)	M_AXIS_MM2S	O	zeros	AXI4-Stream Write Keep Out. Indicates valid bytes on stream data.
m_axis_mm2s_tvalid	M_AXIS_MM2S	O	0	AXI4-Stream Stream Valid Out. Indicates stream data bus, m_axis_mm2s_tdata, is valid • 1 = Write data is valid. • 0 = Write data is not valid.
m_axis_mm2s_tready	M_AXIS_MM2S	I		AXI4-Stream Ready. Indicates to MM2S channel target is ready to receive stream data. • 1 = Ready to receive data. • 0 = Not ready to receive data.
m_axis_mm2s_tlast	M_AXIS_MM2S	O	0	AXI4-Stream Last. Indicates last data beat of stream data. • 1 = Last data beat • 0 = Not last data beat
MM2S Master Control Stream Interface Signals				
mm2s_cntrl_reset_out_n	M_AXIS_CNTRL	O	0	Control Reset Out
m_axis_mm2s_cntrl_tdata (C_M_AXIS_MM2S_CNTRL_TDATA_WIDTH-1: 0)	M_AXIS_CNTRL	O	zeros	AXI Control Stream Stream Data Out
m_axis_mm2s_cntrl_tkeep (C_M_AXIS_MM2S_CNTRL_TDATA_WIDTH/8-1: 0)	M_AXIS_CNTRL	O	zeros	AXI Control Stream Write Keep Out. Indicates valid bytes on stream data.
m_axis_mm2s_cntrl_tvalid	M_AXIS_CNTRL	O	0	AXI Control Stream Stream Valid Out. Indicates stream data bus, m_axis_mm2s_cntrl_tdata, is valid. • 1 = Write data is valid. • 0 = Write data is not valid.
m_axis_mm2s_cntrl_tready	M_AXIS_CNTRL	I		AXI Control Stream Ready. Indicates to MM2S channel target is ready to receive stream data. • 1 = Ready to receive data • 0 = Not ready to receive data
m_axis_mm2s_cntrl_tlast	M_AXIS_CNTRL	O	0	AXI Control Stream Last. Indicates last data beat of stream data. • 1 = Last data beat • 0 = Not last data beat

Table 2-1: **I/O Signal Description *(Cont'd)***

Signal Name	Interface	Signal Type	Init Status	Description
S2MM Memory Map Write Interface Signals				
m_axi_s2mm_awaddr (C_M_AXI_S2MM_ADDR_ WIDTH-1: 0)	M_AXI_S2MM	O	zeros	Write Address Channel Address Bus
m_axi_s2mm_awlen(7: 0)	M_AXI_S2MM	O	zeros	Write Address Channel Burst Length. In data beats - 1.
m_axi_s2mm_awsize(2: 0)	M_AXI_S2MM	O	zeros	Write Address Channel Burst Size. Indicates with of burst transfer. • 000b = 1 byte (8-bit wide burst) • 001b = 2 bytes (16-bit wide burst) • 010b = 4 bytes (32-bit wide burst) • 011b = 8 bytes (64-bit wide burst). • 100b = 16 bytes (128-bit wide burst) • 101b = 32 bytes (256-bit wide burst) • 110b = Not Supported by AXI DMA • 111b = Not Supported by AXI DMA
m_axi_s2mm_awburst(1:0)	M_AXI_S2MM	O	zeros	Write Address Channel Burst Type. Indicates type burst. • 00b = FIXED - Not supported • 01b = INCR - Incrementing address • 10b = WRAP - Not supported • 11b = Reserved
m_axi_s2mm_awprot(2:0)	M_AXI_S2MM	O	010b	Write Address Channel Protection. This is always driven with a constant output of 0010b.
m_axi_s2mm_awcache(3:0)	M_AXI_S2MM	O	0011b	Write Address Channel Cache. This is always driven with a constant output of 0011b.
m_axi_s2mm_awaddr	M_AXI_S2MM	O	0	Write Address Channel Write Address Valid. Indicates if mm2s_axim_awaddr is valid. • 1 = Write Address is valid. • 0 = Write Address is not valid.
m_axi_s2mm_awready	M_AXI_S2MM	I		Write Address Channel Write Address Ready. Indicates target is ready to accept the write address. • 1 = Target ready to accept address. • 0 = Target not ready to accept address.
m_axi_s2mm_wdata (C_M_AXI_S2MM_DATA_ WIDTH-1: 0)	M_AXI_S2MM	O	zeros	Write Data Channel Write Data Bus
m_axi_s2mm_wstrb (C_M_AXI_S2MM_DATA_ WIDTH/8 - 1: 0)	M_AXI_S2MM	O	zeros	Write Data Channel Write Strobe Bus. Indicates which bytes are valid in the write data bus. This value is passed from the stream side strobe bus.

Table 2-1: **I/O Signal Description *(Cont'd)***

Signal Name	Interface	Signal Type	Init Status	Description
m_axi_s2mm_wlast	M_AXI_S2MM	O	0	Write Data Channel Last. Indicates the last data beat of a burst transfer. • 1 = Last data beat • 0 = Not last data beat
m_axi_s2mm_wvalid	M_AXI_S2MM	O	0	Write Data Channel Data Valid. Indicates m_axi_s2mm_wdata is valid. • 1 = Valid write data • 0 = Not valid write data
m_axi_s2mm_wready	M_AXI_S2MM	I		Write Data Channel Ready. Indicates the write channel target is ready to accept write data. • 1 = Target is ready. • 0 = Target is not ready.
m_axi_s2mm_bresp(1:0)	M_AXI_S2MM	I		Write Response Channel Response. Indicates results of the write transfer. • 00b = OKAY - Normal access has been successful. • 01b = EXOKAY - Not supported • 10b = SLVERR - Slave returned error on transfer. • 11b = DECERR - Decode error, transfer targeted unmapped address.
m_axi_s2mm_bvalid	M_AXI_S2MM	I		Write Response Channel Response Valid. Indicates response, m_axi_s2mm_bresp, is valid. • 1 = Response is valid. • 0 = Response is not valid.
m_axi_s2mm_bready	M_AXI_S2MM	O	0	Write Response Channel Ready. Indicates S2MM write channel is ready to receive response. • 1 = Ready to receive response. • 0 = Not ready to receive response.
S2MM Slave Stream Interface Signals				
s2mm_prmry_reset_out_n	S_AXIS_S2MM	O	0	Primary S2MM Reset Out
s_axis_s2mm_tdata (C_S_AXIS_S2MM_TDATA _WIDTH-1: 0)	S_AXIS_S2MM	I		AXI4-Stream Stream Data In
s_axis_s2mm_tkeep (C_S_AXIS_S2MM_TDATA _WIDTH/8-1: 0)	S_AXIS_S2MM	I		AXI4-Stream Write Keep In. Indicates valid bytes on stream data.
s_axis_s2mm_tvalid	S_AXIS_S2MM	I		AXI4-Stream Stream Valid In. Indicates stream data bus, s_axis_s2mm_tdata, is valid. • 1 = Write data is valid. • 0 = Write data is not valid.

Table 2-1: **I/O Signal Description *(Cont'd)***

Signal Name	Interface	Signal Type	Init Status	Description
s_axis_s2mm_tready	S_AXIS_S2MM	O	0	AXI4-Stream Ready. Indicates S2MM channel stream interface ready to receive stream data. • 1 = Ready to receive data • 0 = Not ready to receive data
s_axis_s2mm_tlast	S_AXIS_S2MM	I		AXI4-Stream Last. Indicates last data beat of stream data. • 1 = Last data beat • 0 = Not last data beat
S2MM Slave Status Stream Interface Signals				
s2mm_sts_reset_out_n	S_AXIS_STS	O	0	AXI Status Stream (STS) Reset Output
s_axis_s2mm_sts_tdata (C_S_AXIS_S2MM_STS_ TDATA_WIDTH-1: 0)	S_AXIS_STS	I		AXI Status Stream Stream Data In
s_axis_s2mm_sts_tkeep (C_S_AXIS_S2MM_STS_ TDATA_WIDTH/8-1: 0)	S_AXIS_STS	I		AXI Status Stream Write Keep In. Indicates valid bytes on stream data.
s_axis_s2mm_sts_tvalid	S_AXIS_STS	I		AXI Status Stream Stream Valid In. Indicates stream data bus, s_axis_s2mm_sts_tdata, is valid. • 1 = Write data is valid. • 0 = Write data is not valid.
s_axis_s2mm_sts_tready	S_AXIS_STS	O	0	AXI Status Stream Ready. Indicates S2MM channel stream interface ready to receive stream data. • 1 = Ready to receive data. • 0 = Not ready to receive data.
s_axis_s2mm_sts_tlast	S_AXIS_STS	I		AXI Status Stream Last. Indicates last data beat of stream data. • 1 = Last data beat • 0 = Not last data beat
Scatter Gather Memory Map Read Interface Signals				
m_axi_sg_araddr (C_M_AXI_SG_ADDR_ WIDTH-1: 0)	M_AXI_SG	O	zeros	Scatter Gather (SG) Read Address Channel Address Bus
m_axi_sg_arlen(7: 0)	M_AXI_SG	O	zeros	Scatter Gather Read Address Channel Burst Length. Length in data beats - 1.

Table 2-1: **I/O Signal Description *(Cont'd)***

Signal Name	Interface	Signal Type	Init Status	Description
m_axi_sg_arsize(2: 0)	M_AXI_SG	O	zeros	Scatter Gather Read Address Channel Burst Size. Indicates with of burst transfer. • 000b = Not Supported by AXI DMA SG Engine. • 001b = Not Supported by AXI DMA SG Engine. • 010b = 4 bytes (32-bit wide burst). • 011b = Not Supported by AXI DMA SG Engine. • 100b = Not Supported by AXI DMA SG Engine. • 101b = Not Supported by AXI DMA SG Engine. • 110b = Not Supported by AXI DMA SG Engine. • 111b = Not Supported by AXI DMA SG Engine.
m_axi_sg_arburst(1:0)	M_AXI_SG	O	zeros	Scatter Gather Read Address Channel Burst Type. Indicates type burst. • 00b = FIXED - Not supported • 01b = INCR - Incrementing address • 10b = WRAP - Not supported • 11b = Reserved
m_axi_sg_arprot(2:0)	M_AXI_SG	O	010b	Scatter Gather Read Address Channel Protection. This is always driven with a constant output of 010b.
m_axi_sg_arcache(3:0)	M_AXI_SG	O	0011b	Scatter Gather Read Address Channel Cache. This is always driven with a constant output of 0011b.
m_axi_sg_arvalid	M_AXI_SG	O	0	Scatter Gather Read Address Channel Read Address Valid. Indicates if m_axi_sg_araddr is valid. • 1 = Read Address is valid. • 0 = Read Address is not valid.
m_axi_sg_arready	M_AXI_SG	I		Scatter Gather Read Address Channel Read Address Ready. Indicates target is ready to accept the read address. • 1 = Target ready to accept address • 0 = Target not ready to accept address
m_axi_sg_rdata (C_M_AXI_SG_DATA_ WIDTH-1: 0)	M_AXI_SG	I		Scatter Gather Read Data Channel Read Data

Table 2-1: **I/O Signal Description *(Cont'd)***

Signal Name	Interface	Signal Type	Init Status	Description
m_axi_sg_rresp(1:0)	M_AXI_SG	I		Scatter Gather Read Data Channel Response. Indicates results of the read transfer. • 00b = OKAY - Normal access has been successful. • 01b = EXOKAY - Not supported • 10b = SLVERR - Slave returned error on transfer. • 11b = DECERR - Decode error, transfer targeted unmapped address.
m_axi_sg_rlast	M_AXI_SG	I		Scatter Gather Read Data Channel Last. Indicates the last data beat of a burst transfer. • 1 = Last data beat • 0 = Not last data beat
m_axi_sg_rdata	M_AXI_SG	I		Scatter Gather Read Data Channel Data Valid. Indicates m_sg_aximry_rdata is valid. • 1 = Valid read data • 0 = Not valid read data
m_axi_sg_rready	M_AXI_SG	O	0	Scatter Gather Read Data Channel Ready. Indicates the read channel is ready to accept read data. • 1 = Is ready • 0 = Is not ready
Scatter Gather Memory Map Write Interface Signals				
m_axi_sg_awaddr (C_M_AXI_SG_ADDR_ WIDTH-1: 0)	M_AXI_SG	O	zeros	Scatter Gather Write Address Channel Address Bus
m_axi_sg_awlen(7: 0)	M_AXI_SG	O	zeros	Scatter Gather Write Address Channel Burst Length. Length in data beats - 1.
m_axi_sg_awsize(2: 0)	M_AXI_SG	O	zeros	Scatter Gather Write Address Channel Burst Size. Indicates with of burst transfer,000b = Not Supported by AXI DMA SG Engine • 001b = Not Supported by AXI DMA SG Engine • 010b = 4 bytes (32-bit wide burst), • 011b = Not Supported by AXI DMA SG Engine • 100b = Not Supported by AXI DMA SG Engine • 101b = Not Supported by AXI DMA SG Engine • 110b = Not Supported by AXI DMA SG Engine • 111b = Not Supported by AXI DMA SG Engine
m_axi_sg_awburst(1:0)	M_AXI_SG	O	zeros	Scatter Gather Write Address Channel Burst Type. Indicates type burst. • 00b = FIXED - Not supported • 01b = INCR - Incrementing address • 10b = WRAP - Not supported • 11b = Reserved

Table 2-1: **I/O Signal Description *(Cont'd)***

Signal Name	Interface	Signal Type	Init Status	Description
m_axi_sg_awprot(2:0)	M_AXI_SG	O	010b	Scatter Gather Write Address Channel Protection. This is always driven with a constant output of 010b.
m_axi_sg_awcache(3:0)	M_AXI_SG	O	0011b	Scatter Gather Write Address Channel Cache. This is always driven with a constant output of 0011b.
m_axi_sg_awvalid	M_AXI_SG	O	0	Scatter Gather Write Address Channel Write Address Valid. Indicates if m_axi_sg_awaddr is valid. • 1 = Write Address is valid. • 0 = Write Address is not valid.
m_axi_sg_awready	M_AXI_SG	I		Scatter Gather Write Address Channel Write Address Ready. Indicates target is ready to accept the write address. • 1 = Target ready to accept address. • 0 = Target not ready to accept address.
m_axi_sg_wdata (C_M_AXI_SG_DATA_ WIDTH-1 : 0)	M_AXI_SG	O	zeros	Scatter Gather Write Data Channel Write Data Bus
m_axi_sg_wstrb (C_M_AXI_SG_DATA_ WIDTH/8 - 1: 0)	M_AXI_SG	O	1111b	Scatter Gather Write Data Channel Write Strobe Bus. All bytes always valid.
m_axi_sg_wlast	M_AXI_SG	O	0	Scatter Gather Write Data Channel Last. Indicates the last data beat of a burst transfer. • 1 = Last data beat • 0 = Not last data beat
m_axi_sg_wvalid	M_AXI_SG	O	0	Scatter Gather Write Data Channel Data Valid. Indicates M_SG_AXIMry_WDATA is valid. • 1 = Valid write data • 0 = Not valid write data
m_axi_sg_wready	M_AXI_SG	I		Scatter Gather Write Data Channel Target Ready. Indicates the write channel target is ready to accept write data. • 1 = Target is ready. • 0 = Target is not ready.
m_axi_sg_bresp(1:0)	M_AXI_SG	I		Scatter Gather Write Response Channel Response. Indicates results of the write transfer. • 00b = OKAY - Normal access has been successful. • 01b = EXOKAY - Not supported • 10b = SLVERR - Slave returned error on transfer. • 11b = DECERR - Decode error, transfer targeted unmapped address.

Table 2-1: **I/O Signal Description *(Cont'd)***

Signal Name	Interface	Signal Type	Init Status	Description
m_axi_sg_bvalid	M_AXI_SG	I		Scatter Gather Write Response Channel Response Valid. Indicates response, m_axi_sg_bresp, is valid. • 1 = Response is valid. • 0 = Response is not valid.
m_axi_sg_bready	M_AXI_SG	O	0	Scatter Gather Write Response Channel Ready. Indicates source is ready to receive response. • 1 = Ready to receive response • 0 = Not ready to receive response

Register Space

The AXI DMA core register space for Scatter / Gather Mode (C_INCLUDE_SG = 1) is shown in Table 2-2. The AXI DMA core register space for Simple DMA Mode (C_INCLUDE_SG = 0) is shown in Table 2-3. The AXI DMA Registers are memory-mapped into non-cacheable memory space. This memory space must be aligned on a AXI word (32-bit) boundary.

AXI DMA Register Address Mapping

Table 2-2: **AXI DMA Scatter / Gather Mode Register Address Mapping (C_INCLUDE_SG = 1)**

Address Space Offset[(1)]	Name	Description
00h	MM2S_DMACR	MM2S DMA Control Register
04h	MM2S_DMASR	MM2S DMA Status Register
08h	MM2S_CURDESC	MM2S Current Descriptor Pointer
0Ch	Reserved	N/A
10h	MM2S_TAILDESC	MM2S Tail Descriptor Pointer
14h to 2Ch	Reserved	N/A
30h	S2MM_DMACR	S2MM DMA Control Register
34h	S2MM_DMASR	S2MM DMA Status Register
38h	S2MM_CURDESC	S2MM Current Descriptor Pointer
3Ch	Reserved	N/A
40h	S2MM_TAILDESC	S2MM Tail Descriptor Pointer

Notes:

1. Address Space Offset is relative to C_BASEADDR assignment. C_BASEADDR is defined in the AXI DMA mpd file and set by Xilinx® Platform Studio (XPS).

Table 2-3: **AXI DMA Simple DMA Mode Register Address Mapping (C_INCLUDE_SG = 0)**

Address Space Offset[(1)]	Name	Description
00h	MM2S_DMACR	MM2S DMA Control Register
04h	MM2S_DMASR	MM2S DMA Status Register
08h - 14h	Reserved	N/A
18h	MM2S_SA	MM2S Source Address
1Ch - 24h	Reserved	N/A
28h	MM2S_LENGTH	MM2S Transfer 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
4Ch - 54h	Reserved	N/A
58h	S2MM_LENGTH	S2MM Buffer Length (Bytes)

Notes:

1. Address Space Offset is relative to C_BASEADDR assignment. C_BASEADDR is defined in the AXI DMA mpd file and set by XPS.

Endianess

All registers are in Little Endian format, as shown in Figure 2-1.

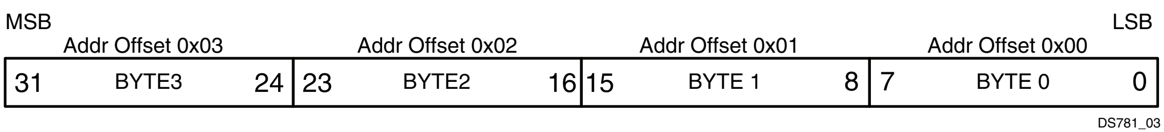


Figure 2-1: **32-bit Little Endian Example**

Memory Map to Stream Register Detail

MM2S_DMACR (MM2S DMA Control Register - Offset 00h) (C_INCLUDE_SG = 1/0)

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

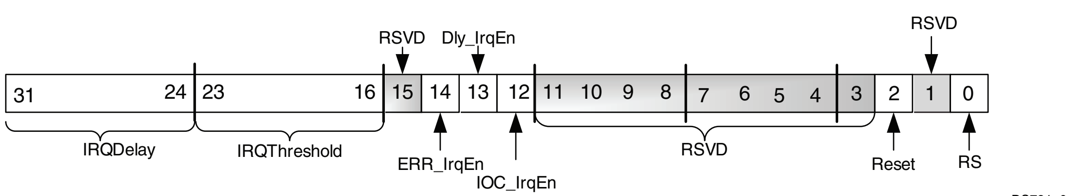


Figure 2-2: **MM2S DMACR Register**

Table 2-4: **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 (C_INCLUDE_SG = 1) 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 Simple DMA Mode (C_INCLUDE_SG = 0) pending commands/transfers are flushed or completed. AXI4-Stream outs are potentially terminated early. 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	RW	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 = Reset NOT in progress - Normal operation. • 1 = Reset in progress.
11 to 3	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 (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
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 Simple DMA Mode (C_INCLUDE_SG = 0)
14	Err_IrqEn	0	R/W	Interrupt on Error Interrupt Enable. When set to 1, allows DMASR.Err_Irq 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.

Table 2-4: **MM2S_DMACR Register Details *(Cont'd)***

Bits	Field Name	Default Value	Access Type	Description
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:*** This field is ignored when AXI DMA is configured for Simple DMA Mode (C_INCLUDE_SG = 0)
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. This is used for cases when the interrupt threshold is not met after a period of time, and the CPU desires an interrupt to be generated. Timer begins counting at the end of a packet and resets with receipt of a new packet or a timeout event occurs. ***Note:*** Setting this value to zero disables the delay timer interrupt. ***Note:*** This field is ignored when AXI DMA is configured for Simple DMA Mode (C_INCLUDE_SG = 0)

Notes:
1. RO = Read Only. Writing has no effect.
2. R/W = Read and Write Accessible

MM2S_DMASR (MM2S DMA Status Register- Offset 04h) (C_INCLUDE_SG = 1/0)

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

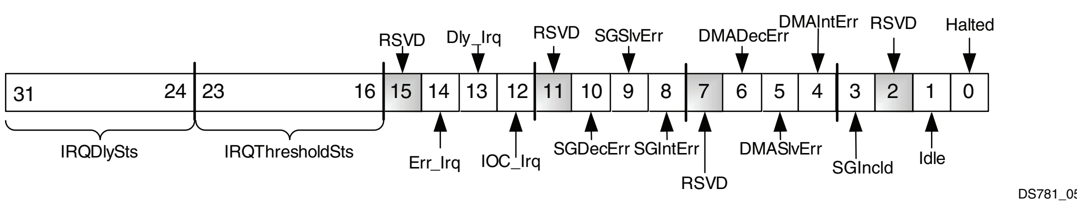


Figure 2-3: **MM2S DMASR Register**

.

Table 2-5: **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 (C_INCLUDE_SG = 1) this bit gets set when DMACR.RS = 0 and DMA and SG operations have halted. For Simple DMA 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 CURDESC_PTR or TAILDESC_PTR pointer registers has no effect on DMA operations when in Scatter Gather Mode (C_INCLUDE_SG = 1). For Simple DMA Mode (C_INCLUDE_SG = 0), 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 (C_INCLUDE_SG = 1) 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 Simple DMA Mode (C_INCLUDE_SG = 0) 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 Simple DMA Mode, transfer is not complete. • 1 = Idle. For Scatter / Gather Mode, SG has reached tail descriptor pointer and DMA operation paused. for Simple DMA 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 Simple DMA mode (C_INCLUDE_SG = 0).
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 Simple DMA mode.

Table 2-5: **MM2S_DMASR Register Details *(Cont'd)***

Bits	Field Name	Default Value	Access Type	Description
4	DMAIntErr	0	RO	DMA Internal Error. Internal error detected by primary AXI DataMover. This error can occur if a 0 length bytes to transfer is fed to the AXI DataMover. This situation 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. ***Note:*** In Scatter / Gather Mode (C_INCLUDE_SG = 1) the CURDESC_PTR register is updated with the errored descriptor pointer when this error is detected. If multiple errors are detected, the errors are logged in the DMASR, but only one address is updated to the CURDESC_PTR. A reset (soft or hard) must be issued to clear the error condition. ***Note:*** This bit is not used and is fixed at 0 when AXI DMA is configured for Simple DMA Mode (C_INCLUDE_SG = 0).
5	DMASlvErr	0	RO	DMA Slave Error. Slave error detected by primary AXI DataMover. 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. ***Note:*** In Scatter / Gather Mode (C_INCLUDE_SG = 1) the CURDESC_PTR register is updated with the errored descriptor pointer when this error is detected. If multiple errors are detected, the errors are logged in the DMASR, but only one address is updated to the CURDESC_PTR. A reset (soft or hard) must be issued to clear the error condition.
6	DMADecErr	0	RO	DMA Decode Error. Decode error detected by primary AXI DataMover. This error occurs if the address request is to an invalid address (that is, 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. ***Note:*** In Scatter / Gather Mode (C_INCLUDE_SG = 1) the CURDESC_PTR register is updated with the errored descriptor pointer when this error is detected. If multiple errors are detected, the errors are logged in the DMASR, but only one address are updated to the CURDESC_PTR. A reset (soft or hard) must be issued to clear the error condition.
7	Reserved	0	RO	Writing to this bit has no effect, and it is always read as zeros.

Table 2-5: **MM2S_DMASR Register Details *(Cont'd)***

Bits	Field Name	Default Value	Access Type	Description
8	SGIntErr	0	RO	Scatter Gather Internal Error. Internal error detected by Scatter Gather AXI DataMover. 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 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:*** The CURDESC_PTR register is updated with the errored descriptor pointer when this error is detected. If multiple errors are detected, the errors are logged in the DMASR, but only one address is updated to the CURDESC_PTR. A reset (soft or hard) must be issued to clear the error condition. ***Note:*** This bit is not used and is fixed at 0 when AXI DMA is configured for Simple DMA Mode (C_INCLUDE_SG = 0).
9	SGSlvErr	0	RO	Scatter Gather Slave Error. Slave error detected by Scatter Gather AXI DataMover. 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:*** The CURDESC_PTR register is updated with the errored descriptor pointer when this error is detected. If multiple errors are detected, the errors are logged in the DMASR, but only one address is updated to the CURDESC_PTR. A reset (soft or hard) must be issued to clear the error condition. ***Note:*** This bit is not used and is fixed at 0 when AXI DMA is configured for Simple DMA Mode (C_INCLUDE_SG = 0).
10	SGDecErr	0	RO	Scatter Gather Decode Error. Decode Error detected by the Scatter Gather AXI DataMover. This error occurs if the address request is to an invalid address (that is, 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:*** The CURDESC_PTR register is updated with the errored descriptor pointer when this error is detected. If multiple errors are detected, the errors are logged in the DMASR, but only one address is updated to the CURDESC_PTR. A reset (soft or hard) must be issued to clear the error condition. ***Note:*** This bit is not used and is fixed at 0 when AXI DMA is configured for Simple DMA Mode (C_INCLUDE_SG = 0).
11	Reserved	0	RO	Writing to this bit has no effect, and it is always read as zeros.

Table 2-5: **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 (C_INCLUDE_SG = 1), indicates an interrupt event was generated on completion of a descriptor. This occurs for descriptors with the EOF bit set. When set to 1 for Simple DMA Mode (C_INCLUDE_SG = 0), indicates an interrupt event was generated on completion of a transfer. This occurs after a packet has completed transfer. If 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.
13	Dly_Irq	0	R/WC	Interrupt on Delay. When set to 1, indicates an interrupt event was generated on delay timer timeout. If enabled (Dly_IrqEn = 1), an interrupt out is generated from the AXI DMA. • 0 = No Delay Interrupt. • 1 = Delay Interrupt detected. ***Note:*** This bit is not used and is fixed at 0 when AXI DMA is configured for Simple DMA Mode (C_INCLUDE_SG = 0).
14	Err_Irq	0	R/WC	Interrupt on Error. When set to 1, indicates an interrupt event was generated on error. If enabled (Err_IrqEn = 1), an interrupt out is generated from the AXI DMA. 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. ***Note:*** This field is not used and is fixed to zeros when AXI DMA is configured for Simple DMA Mode (C_INCLUDE_SG = 0).
31 to 24	IRQDelaySts	00h	RO	Interrupt Delay Time Status. Indicates current interrupt delay time value. ***Note:*** This field is not used and is fixed to zeros when AXI DMA is configured for Simple DMA Mode (C_INCLUDE_SG = 0).

Notes:

1. RO = Read Only. Writing has no effect.
2. R/WC = Read / Write to Clear. A CPU write of 1 clears the associated bit to 0.

MM2S_CURDESC (MM2S DMA Current Descriptor Pointer Register- Offset 08h) (C_INCLUDE_SG = 1)

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

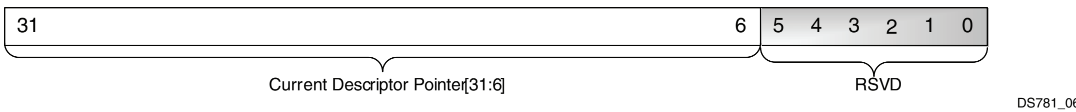


Figure 2-4: **MM2S CURDESC Register**

Table 2-6: **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)(1)	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.

Notes:

1. RO = Read Only. Writing has no effect when channel is running.
2. R/W = Read and Write accessible.

MM2S_TAILDESC (MM2S DMA Tail Descriptor Pointer Register- Offset 10h) (C_INCLUDE_SG = 1)

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

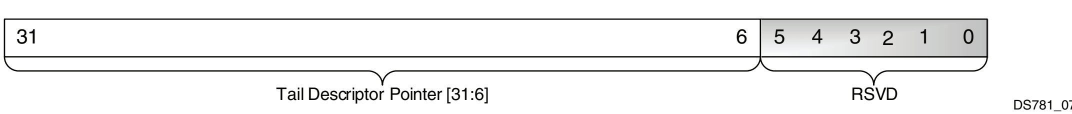


Figure 2-5: **MM2S_TAILDESC Register**

Table 2-7: **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.
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, 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.

Notes:

1. RO = Read Only. Writing has no effect.
2. R/W = Read and Write accessible.

MM2S_SA (MM2S DMA Source Address Register- Offset 18h) (C_INCLUDE_SG = 0)

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

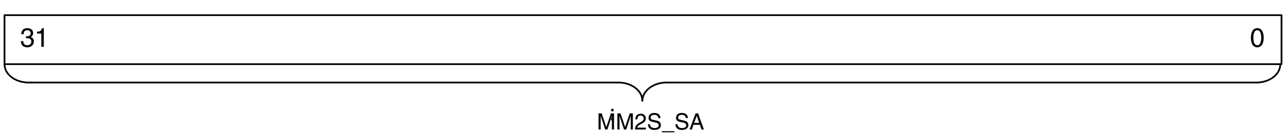


Figure 2-6: **MM2S_SA Register**

Table 2-8: **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 (C_INCLUDE_MM2S_DRE = 1), the Source Address can be at any byte offset. If Data Realignment Engine is not included (C_INCLUDE_MM2S_DRE = 0), the Source Address must be S2MM stream data width aligned.

Notes:

1. R/W = Read and Write accessible.

MM2S_LENGTH (MM2S DMA Transfer Length Register- Offset 28h) (C_INCLUDE_SG = 0)

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

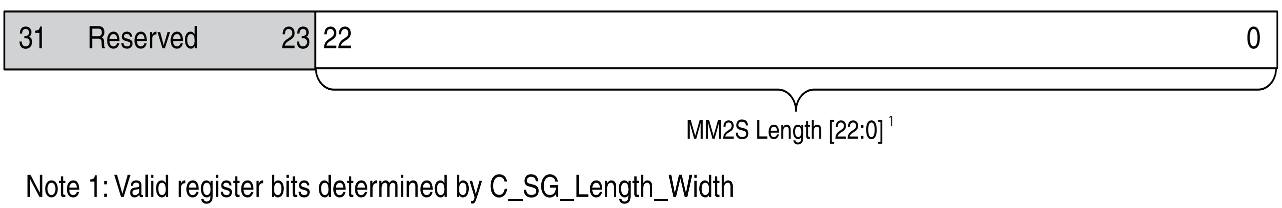


Figure 2-7: **MM2S_LENGTH Register**

Table 2-9: **MM2S_LENGTH Register Details**

Bits	Field Name	Default Value	Access Type	Description
22[2] 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 23	Reserved	0	RO	Writing to these bits has no effect and they are always read as zeros.

Notes:

1. R/W = Read and Write accessible.
2. Width of Length field determined by C_SG_LENGTH_WIDTH parameter. Minimum width is 8 bits (7 to 0) and maximum width is 23 bits (22 to 0).

Stream to Memory Map Register Detail

S2MM_DMACR (S2MM DMA Control Register - Offset 30h) (C_INCLUDE_SG = 1/0)

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

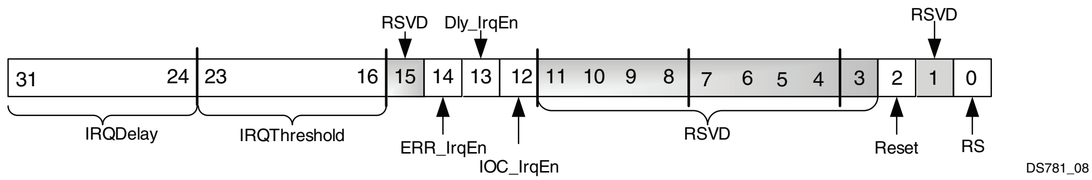


Figure 2-8: **S2MM DMACR Register**

Table 2-10: **S2MM_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 (C_INCLUDE_SG = 1) 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 Simple DMA Mode (C_INCLUDE_SG = 0) pending commands/transfers are flushed or completed. AXI4-Streams are potentially terminated early. 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.
11 to 3	Reserved	0	RO	Writing to these bits has no effect and they are always read as zeros.

Table 2-10: **S2MM_DMACR Register Details *(Cont'd)***

Bits	Field Name	Default Value	Access Type	Description
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 IOC bit set. • 0 = IOC Interrupt disabled. • 1 = IOC Interrupt enabled.
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:*** This bit is ignored when the AXI DMA is configured for Simple DMA Mode (C_INCLUDE_SG = 0).
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:*** This field is ignored when the AXI DMA is configured for Simple DMA Mode (C_INCLUDE_SG = 0)
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. This is used for cases when the interrupt threshold is not met after a period of time and the CPU desires an interrupt to be generated. The timer begins counting at the end of a packet and resets with the receipt of a new packet or a timeout event occurs. ***Note:*** Setting this value to zero disables the delay timer interrupt. ***Note:*** This field is ignored when AXI DMA is configured for Simple DMA Mode (C_INCLUDE_SG = 0)

Notes:

1. RO = Read Only. Writing has no effect.
2. R/W = Read and Write Accessible.

S2MM_DMASR (S2MM DMA Status Register- Offset 34h) (C_INCLUDE_SG = 1/0)

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

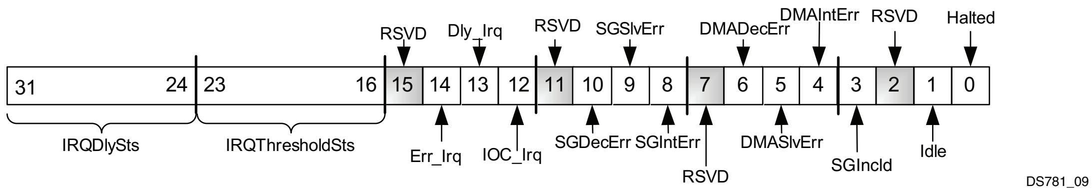


Figure 2-9: **S2MM DMASR Register**

Table 2-11: **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 (C_INCLUDE_SG = 1) this bit gets set when DMACR.RS = 0 and DMA and SG operations have halted. For Simple DMA 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 CURDESC_PTR or TAILDESC_PTR pointer registers has no effect on DMA operations when in Scatter Gather Mode (C_INCLUDE_SG = 1). For Simple DMA Mode (C_INCLUDE_SG = 0), 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 (C_INCLUDE_SG = 1) 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 Simple DMA Mode (C_INCLUDE_SG = 0) 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 Simple DMA Mode, transfer is not complete. • 1 = Idle. For Scatter / Gather Mode, SG has reached tail descriptor pointer and DMA operation paused. For Simple DMA 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 is configured for Simple DMA mode (C_INCLUDE_SG = 0).
2	Reserved	0	RO	Writing to this bit has no effect and it is always read as zero.

Table 2-11: **S2MM_DMASR Register Details *(Cont'd)***

Bits	Field Name	Default Value	Access Type	Description
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 Simple DMA mode.
4	DMAIntErr	0	RO	DMA Internal Error. Internal Error detected by primary AXI DataMover. This error can occur if a 0 length bytes to transfer is fed to the AXI DataMover. This happens if the buffer length specified in the fetched descriptor is set to 0. Also, when in Scatter Gather Mode, C_INCLUDE_SG = 1 and using the status app length field, C_SG_INCLUDE_STSCNTRL_STRM = 1 and C_SG_USE_STSAPP_LEGNTH = 1, this error occurs when the Status AXI4-Stream packet's RxLength field does not match the S2MM packet being received by the S_AXIS_S2MM interface. 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:*** In Scatter / Gather Mode (C_INCLUDE_SG = 1) the CURDESC_PTR register is updated with the errored descriptor pointer when this error is detected. If multiple errors are detected, the errors are logged in the DMASR, but only one address is updated to the CURDESC_PTR. A reset (soft or hard) must be issued to clear the error condition. ***Note:*** This bit is not used and is fixed at 0 when AXI DMA is configured for Simple DMA Mode (C_INCLUDE_SG = 0).
5	DMASlvErr	0	RO	DMA Slave Error. Slave Error detected by primary AXI DataMover. 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. ***Note:*** In Scatter / Gather Mode (C_INCLUDE_SG = 1) the CURDESC_PTR register is updated with the errored descriptor pointer when this error is detected. If multiple errors are detected, the errors are logged in the DMASR, but only one address is updated to the CURDESC_PTR. A reset (soft or hard) must be issued to clear the error condition.

Table 2-11: **S2MM_DMASR Register Details *(Cont'd)***

Bits	Field Name	Default Value	Access Type	Description
6	DMADecErr	0	RO	DMA Decode Error. Decode error detected by primary AXI DataMover. This error occurs if the address request is to an invalid address (that is, 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. ***Note:*** In Scatter / Gather Mode (C_INCLUDE_SG = 1) the CURDESC_PTR register is updated with the errored descriptor pointer when this error is detected. If multiple errors are detected, the errors are logged in the DMASR, but only one address is updated to the CURDESC_PTR. A reset (soft or hard) must be issued to clear the error condition.
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. Internal Error detected by Scatter Gather AXI DataMover. 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. DMA Engine halts. ***Note:*** The CURDESC_PTR register is updated with the errored descriptor pointer when this error is detected. A reset (soft or hard) must be issued to clear the error condition. This error cannot be logged into the descriptor. ***Note:*** This bit is not used and is fixed at 0 when AXI DMA is configured for Simple DMA Mode (C_INCLUDE_SG = 0).
9	SGSlvErr	0	RO	Scatter Gather Slave Error. Slave error detected by Scatter Gather AXI DataMover. 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. ***Note:*** The CURDESC_PTR register is updated with the errored descriptor pointer when this error is detected. If multiple errors are detected, the errors are logged in the DMASR, but only one address is updated to the CURDESC_PTR. A reset (soft or hard) must be issued to clear the error condition. This error cannot be logged into the descriptor. ***Note:*** This bit is not used and is fixed at 0 when AXI DMA is configured for Simple DMA Mode (C_INCLUDE_SG = 0).

Table 2-11: **S2MM_DMASR Register Details *(Cont'd)***

Bits	Field Name	Default Value	Access Type	Description
10	SGDecErr	0	RO	Scatter Gather Decode Error. Decode Error detected by the Scatter Gather AXI DataMover. This error occurs if the address request is to an invalid address (that is, 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:*** The CURDESC_PTR register is updated with the errored descriptor pointer when this error is detected. If multiple errors are detected, the errors are logged in the DMASR, but only one address is updated to the CURDESC_PTR. A reset (soft or hard) must be issued to clear the error condition. This error cannot be logged into the descriptor. ***Note:*** This bit is not used and is fixed at 0 when AXI DMA is configured for Simple DMA Mode (C_INCLUDE_SG = 0).
11	Reserved	0	RO	Writing to this bit has no effect and it is always read as zeros.
12	IOC_Irq	0	R/WC	Interrupt on Complete. When set to 1 for Scatter / Gather Mode (C_INCLUDE_SG = 1) indicates an interrupt event was generated on completion of a descriptor. This occurs for descriptors with the EOF bit set. When set to 1 for Simple DMA Mode (C_INCLUDE_SG = 0) indicates an interrupt event was generate on completion of a transfer. This occurs after a packet has completed transfer. If 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.
13	Dly_Irq	0	R/WC	Interrupt on Delay. When set to 1, indicates an interrupt event was generated on delay timer timeout. If enabled (Dly_IrqEn = 1), an interrupt out is generated from the AXI DMA. • 0 = No Delay Interrupt. • 1 = Delay Interrupt detected. ***Note:*** This bit is not used and is fixed at 0 when the AXI DMA is configured for Simple DMA Mode (C_INCLUDE_SG = 0).
14	Err_Irq	0	R/WC	Interrupt on Error. When set to 1, indicates an interrupt event was generated on error. If enabled (Err_IrqEn = 1), an interrupt out is generated from the AXI DMA. • 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. ***Note:*** This field is not used and is fixed to zeros when the AXI DMA is configured for Simple DMA Mode (C_INCLUDE_SG = 0).

Table 2-11: **S2MM_DMASR Register Details *(Cont'd)***

Bits	Field Name	Default Value	Access Type	Description
31 to 24	IRQDelaySts	00h	RO	Interrupt delay time Status. Indicates current interrupt delay time value. ***Note:*** This field is not used and is fixed to zeros when the AXI DMA is configured for Simple DMA Mode (C_INCLUDE_SG = 0).

Notes:

1. RO = Read Only. Writing has no effect.
2. R/WC = Read / Write to Clear. A CPU write of 1 clears the associated bit to 0.

S2MM_CURDESC (S2MM DMA Current Descriptor Pointer Register- Offset 38h) (C_INCLUDE_SG = 1)

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

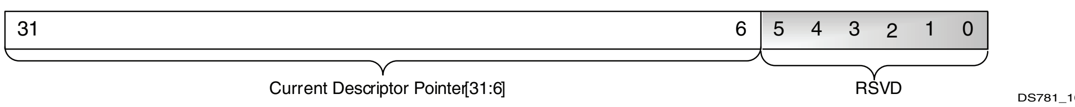


Figure 2-10: **S2MM CURDESC Register**

Table 2-12: **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)(2)	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). Buffer Descriptors must be 16-word aligned, that is, 0x00, 0x40, 0x80, and so forth. Any other alignment has undefined results.

Notes:

1. RO = Read Only. Writing has no effect when engine is running.
2. R/W = Read and Write Accessible.

S2MM_TAILDESC (S2MM DMA Tail Descriptor Pointer Register- Offset 40h) (C_INCLUDE_SG = 1)

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

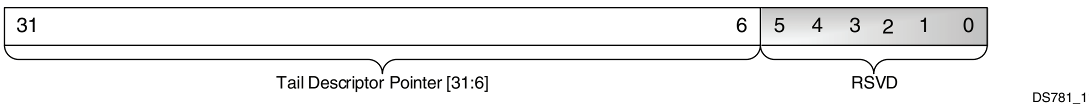


Figure 2-11: **S2MM TAILDESC Register**

Table 2-13: **S2MM_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.
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 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.

Notes:
1. RO = Read Only. Writing has no effect.
2. R/W = Read and Write Accessible

S2MM_DA (MM2S DMA Destination Address Register- Offset 48h) (C_INCLUDE_SG = 0)

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

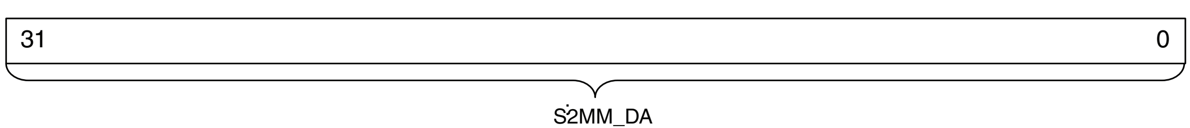


Figure 2-12: **S2MM_DA Register**

Table 2-14: **S2MM_DA Register Details**

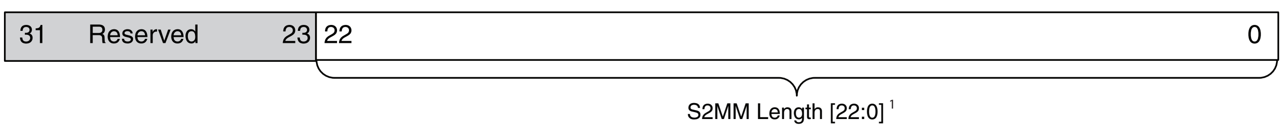
Bits	Field Name	Default Value	Access Type	Description
31 to 0	Destination Address	zeros	R/W	Indicates the source address the AXI DMA reads from to transfer data to AXI4-Stream on S2MM Channel. ***Note:*** If Data Realignment Engine is included (C_INCLUDE_S2MM_DRE = 1), the Destination Address can be at any byte offset. If Data Realignment Engine is not included (C_INCLUDE_S2MM_DRE = 0), the Destination Address must be S2MM stream data width aligned.

Notes:

1. R/W = Read and Write accessible.

S2MM_LENGTH (S2MM DMA Buffer Length Register- Offset 58h) (C_INCLUDE_SG = 0)

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



Note 1: Valid register bits determined by C_SG_Length_Width

Figure 2-13: **S2MM_LENGTH Register**

Table 2-15: **S2MM_LENGTH Register Details**

Bits	Field Name	Default Value	Access Type	Description
22[(2)] 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 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 23	Reserved	0	RO	Writing to these bits has no effect and they are always read as zeros.

Notes:

1. R/W = Read and Write accessible.
2. Width of Length field determined by C_SG_LENGTH_WIDTH parameter. Minimum width is 8 bits (7 to 0) and maximum width is 23 bits (22 to 0).

Chapter 3

Customizing and Generating the Core

This chapter includes information on using Xilinx tools to customize and generate the core.

Generating the Core

The AXI DMA can be found in **AXI_Infrastructure** and also in **Communication_&_Networking\Ethernet** in the CORE Generator™ tool graphical user interface (GUI) View by Function pane.

To access the AXI DMA, do the following:

1. Open a project by selecting **File** then **Open Project** or create a new project by selecting **File** then **New Project**.
2. With an open project, choose **AXI_Infrastructure** in the **View by Function** pane.
3. Double-click on **AXI Direct Memory Access** to bring up the AXI DMA GUI.

CORE Generator Software Parameter Screen

The AXI DMA GUI contains one screen (Figure 3-1) providing information about the core, allows for configuration of the core, and provides the ability to generate the core.

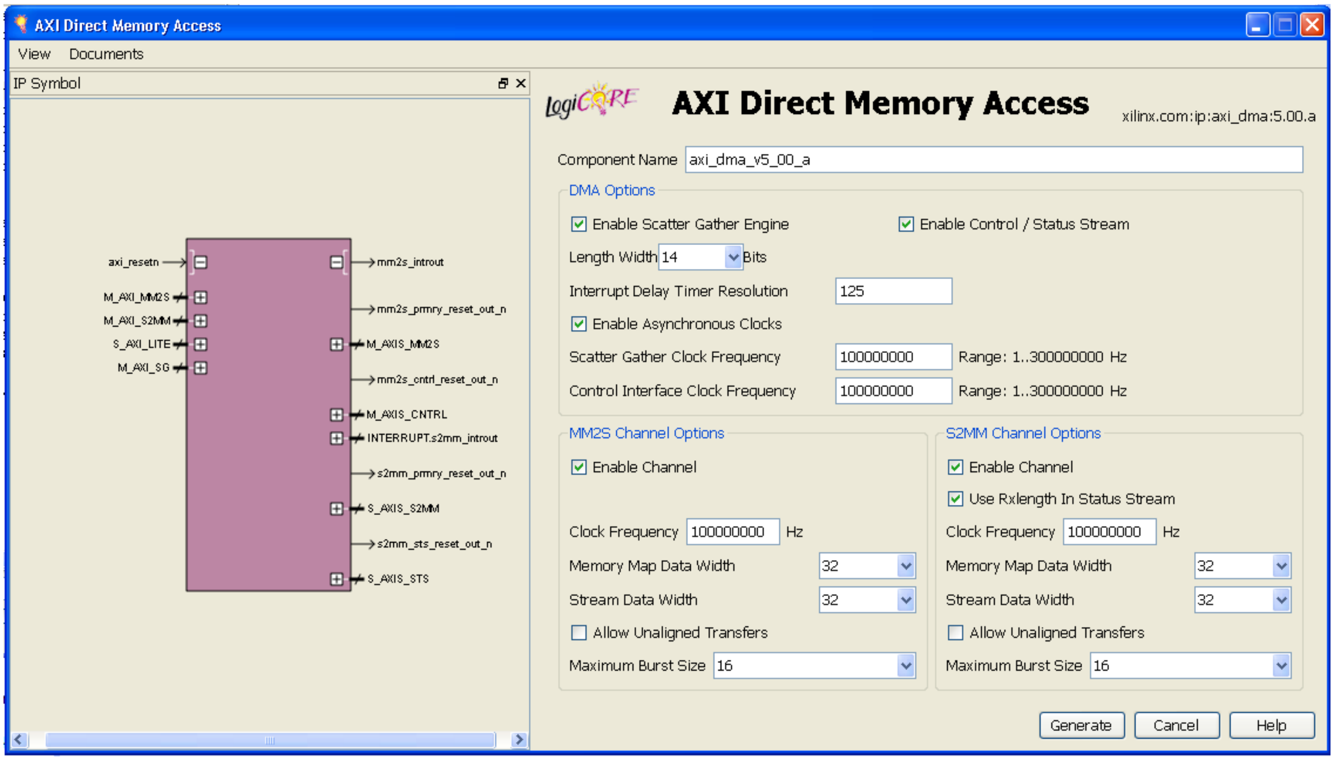


Figure 3-1: **AXI DMA GUI**

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 "_".

DMA Options

The following describe options that affect both channels of the AXI DMA core.

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 Simple DMA 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 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 `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 is indicated by a Receive End Of Frame (RXEOF = 1) in the status word of the updated descriptor.

Note: This feature requires the inclusion of the Scatter Gather Engine.

Length Width

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 Status Stream App4 field when Use Rxlength is enabled. For Simple DMA 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 $2^{\text{Length Width}}$. So a Length Width of 23 gives a byte count of 8388608 Bytes or 8 MegaBytes.

Interrupt Delay Timer Resolution

This integer value sets the resolution of the Interrupt Delay Counter. Values specify the number of clock cycles between each tick of the delay counter. If Scatter Gather Engine is enabled then clock cycles are based on the `m_axi_sg_aclk` clock input. When in Simple DMA Mode, that is Scatter Gather Engine is disabled, then the delay interrupt is not used by AXI DMA.

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 Asynchronous Clocks are disabled, all clocks must be at the same frequency and from the same source.

Scatter Gather Clock Frequency

This setting specifies the clock frequency in hertz of the Scatter Gather Engine interface clock, `m_axi_sg_aclk`. This parameter is used when Asynchronous Clocks are enabled and configures the AXI DMA for proper clock domain crossings. When Asynchronous Clocks are disabled, this setting is ignored by AXI DMA.

Control Interface Clock Frequency

This setting specifies the clock frequency in hertz of the AXI4-Lite Control interface clock, `s_axi_lite_aclk`. This parameter is used when Asynchronous Clocks are enabled and configures the AXI DMA for proper clock domain crossings. When Asynchronous Clocks are disabled, this setting is ignored by AXI DMA.

MM2S Channel Options

The following describe options that 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.

Clock Frequency

This setting specifies the clock frequency in hertz of the MM2S interface clock, `m_axi_mm2s_aclk`. This parameter is used when Asynchronous Clocks are enabled and configures the AXI DMA for proper clock domain crossings. When Asynchronous Clocks are disabled, this setting is ignored by AXI DMA.

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 1024.

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 1024.

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. What is considered aligned or unaligned is based on the stream data width C_M_AXIS_MM2S_TDATA_WIDTH. For example, if C_M_AXIS_MM2S_TDATA_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 C_M_AXIS_MM2S_TDATA_WIDTH = 64, data is aligned if it is located at double-word offsets (64-bit offsets), that is 0x0, 0x8, 0x10, 0x18, and so forth.

For use cases where all transfers are C_M_AXIS_MM2S_TDATA_WIDTH aligned, DRE can be disabled excluding DRE and saving FPGA resources.

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 64-bits and under.

Maximum Burst Size

This option specifies the maximum size of the burst cycles on the AXI4 MM2S Memory Map Read interface. In other words, this setting specifies the granularity of burst partitioning. For example, if the burst length is set to 16, the maximum burst on the memory map interface is 16 data beats. Smaller values reduce throughput but result in less impact on the AXI infrastructure. Larger values increase throughput but result in a greater impact on the AXI infrastructure. Valid values are 16, 32, 64, 128, and 256.

S2MM Channel Options

The following describe options that 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 AXI DMA core. Outputs for S2MM channel are tied to zero and inputs are ignored by AXI DMA.

Use RxLength In Status Stream

If the Control / Status Stream is enabled, then checking this allows AXI DMA to use a receive length field that is supplied by the S2MM target IP in App4 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 provides for a higher bandwidth solution for systems needing greater throughput. It should be noted, that in this configuration the AXI DMA removes the S2MM store and forward feature so care must be taken by the user to make sure the S2MM target IP can supply all data bytes specified in the receive length field of status packet APP4.

Clock Frequency

This setting specifies the clock frequency in hertz of the S2MM interface clock, `m_axi_s2mm_aclk`. This parameter is used when Asynchronous Clocks are enabled and configures the AXI DMA for proper clock domain crossings. When Asynchronous Clocks are disabled, this setting is ignored by AXI DMA.

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 1024.

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 1024.

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 then 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. What is considered aligned or unaligned is based on the stream data width C_S_AXIS_S2MM_TDATA_WIDTH. For example, if C_S_AXIS_S2MM_TDATA_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 C_S_AXIS_S2MM_TDATA_WIDTH = 64, data is aligned if it is located at double-word offsets (64-bit offsets), that is 0x0, 0x8, 0x10, 0x18, and so forth.

For use cases where all transfers are C_S_AXIS_S2MM_TDATA_WIDTH aligned, DRE can be disabled excluding DRE and saving FPGA resources.

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 64-bits and under.

Maximum Burst Size

This setting specifies the maximum size of the burst cycles on the AXI S2MM Memory Map Write interface. In other words, this setting specifies the granularity of burst partitioning. For example, if the burst length is set to 16, the maximum burst on the memory map interface is 16 data beats. Smaller values reduce throughput but result in less impact on the AXI infrastructure. Larger values increase throughput but result in a greater impact on the AXI infrastructure. Valid values are 16, 32, 64, 128, and 256.

Core Implementation

Functional Simulation

VHDL source files for axi_dma_v5_00_a are provided un-encrypted for use in behavioral simulation within a simulation environment. Neither a test bench nor test fixture is provided with the AXI DMA core.

Synthesis

Synthesis of the AXI DMA can be performed with Xilinx® Synthesis Technology (XST).

Xilinx Tools

See the LogiCORE™ IP Facts table.

Static Timing Analysis

Static timing analysis can be performed using *trce*, following *ngdbuild*, *map*, and *par.*

Output Generation

Figure 3-2 shows the output hierarchy that is generated.

Figure 3-2: **Hierarchy**

- \doc contains readme.txt, changelog.html, vinfo.html, product guide and other files.
- \vhdl contains hdl rtl files

Chapter 4

Designing with the Core

This chapter includes guidelines and additional information to make designing with the core easier.

Design Parameters

The AXI DMA Design Parameters are listed and described in Table 4-1.

Table 4-1: **Design Parameter Description**

Feature/Description	Parameter Name	Allowable Values	Default Values	VHDL Type
AXI DMA General Parameters				
Data width in bits of AXI4-Lite Interface	C_S_AXI_LITE_ DATA_WIDTH	32	32	integer
Address width in bits of AXI4-Lite Interface	C_S_AXI_LITE_ ADDR_WIDTH	32	32	integer
Resolution of the interrupt delay timer in axi_scndry_aclk cycles	C_DLYTMR_RESOLUTION	1 - 1000000	125	integer
Primary clock is asynchronous.	C_PRMRY_IS_ACLK_ ASYNC	0,1	0	integer
Frequency in hertz of the s_axi_lite_aclk clock input. This parameter is automatically set by the EDK Tool Suite.	C_S_AXI_LITE_ACLK_ FREQ_HZ		100000000	integer
Frequency in hertz of the m_axi_sg_aclk clock input. This parameter is automatically set by the EDK Tool Suite.	C_M_AXI_SG_ACLK_ FREQ_HZ		100000000	integer
Frequency in hertz of the m_axi_mm2s_aclk clock input. This parameter is automatically set by the EDK Tool Suite.	C_M_AXI_MM2S_ACLK_ FREQ_HZ		100000000	integer
Frequency in hertz of the m_axi_s2mm_aclk clock input. This parameter is automatically set by the EDK Tool Suite.	C_M_AXI_S2MM_ACLK_ FREQ_HZ		100000000	integer

Table 4-1: **Design Parameter Description *(Cont'd)***

Feature/Description	Parameter Name	Allowable Values	Default Values	VHDL Type
Specifies the target Field Programmable Gate Array (FPGA) family	C_FAMILY	virtex6, spartan6, virtex7, kintex7	virtex6	String
Scatter Gather Engine Parameters				
Include or Exclude Scatter Gather Engine • 0 = Exclude Scatter Gather Engine. Enables Simple DMA mode. • 1 = Include Scatter Gather Engine. Enables Scatter/Gather Mode.	C_INCLUDE_SG	0,1	1	Integer
Data width of AXI Scatter Gather Engine	C_M_AXI_SG_DATA_ WIDTH	32	32	integer
Address width of AXI Scatter Gather Engine	C_M_AXI_SG_ADDR_ WIDTH	32	32	integer
Include or Exclude Descriptor Queuing • 0 = Exclude Descriptor Queue • 1 = Include Descriptor Queue	C_SG_INCLUDE_DESC_ QUEUE	0,1	0	integer
Include or Exclude Control and Status Streams • 0 = Exclude Status and Control Streams • 1 = Include Status and Control Streams	C_SG_INCLUDE_ STSCNTRL_STRM	0,1	1	integer
Enable use of receive length in Status Stream Application (APP) Field	C_SG_USE_STSAPP_ LENGTH	0,1	1	integer
Width of the Buffer Length and Transferred Bytes fields as well as receive length value in the status stream application word	C_SG_LENGTH_WIDTH	8 to 23	14	integer
AXI Control Stream Data Width	C_M_AXIS_MM2S_CNTRL _TDATA_WIDTH	32	32	integer
AXI Status Stream Data Width	C_S_AXIS_S2MM_STS_ TDATA_WIDTH	32	32	integer

Table 4-1: **Design Parameter Description *(Cont'd)***

Feature/Description	Parameter Name	Allowable Values	Default Values	VHDL Type
Memory Map to Stream Parameters				
Include or exclude the Memory Map to Stream channel. When excluded, all unused ports are tied off or driven to zero. This also excludes MM2S AXI Control Stream. • 0 = Exclude MM2S • 1 = Include MM2S	C_INCLUDE_MM2S	0,1	1	integer
Include or exclude the Memory Map to Stream channel Data Realignment Engine (DRE). • 0 = Exclude DRE • 1 = Include DRE ***Note:*** DRE support is not available for AXI4-Stream data widths of 128 bits and 256 bits.	C_INCLUDE_MM2S_DRE	0,1	0	integer
Address width of AXI4 Memory Map on the Memory Map to Stream interface	C_M_AXI_MM2S_ADDR_ WIDTH	32	32	integer
Data width of AXI4 Memory Map on the Memory Map to Stream Interface	C_M_AXI_MM2S_DATA_ WIDTH	32,64,128, 256, 512, 1024	32	integer
Data width of AXI4-Stream on the Stream to Memory Map Interface. Width must be equal or less than C_M_AXI_MM2S_DATA_WIDTH .	C_M_AXIS_MM2S_TDATA _WIDTH	8,16, 32,64,128, 256,512, 1024	32	integer
Maximum burst size per burst request on Memory Map Read interface	C_MM2S_BURST_SIZE	16,32,64,1 28, 256, 512,1024	16	integer
Stream to Memory Map Parameters				
Include or exclude the Stream to Memory Map. When excluded, all unused ports are tied off or driven to zero. This also excludes S2MM AXI Status Stream. • 0 = Exclude S2MM • 1 = Include S2MM	C_INCLUDE_S2MM	0,1	1	integer

Table 4-1: **Design Parameter Description *(Cont'd)***

Feature/Description	Parameter Name	Allowable Values	Default Values	VHDL Type
Include or exclude the Stream to Memory Map channel Data Realignment Engine. • 0 = Exclude DRE • 1 = Include DRE ***Note:*** DRE support is not available for AXI4-Stream data widths of 128 bits and 256 bits.	C_INCLUDE_S2MM_DRE	0,1	0	integer
Address width of AXI4 Memory Map on the Stream to Memory Map interface	C_M_AXI_S2MM_ADDR_ WIDTH	32	32	integer
Data width of AXI4 Memory Map on the Stream to Memory Map Interface	C_M_AXI_S2MM_DATA_ WIDTH	32,64,128, 256,512, 1024	32	integer
Data width of AXI4-Stream on the Stream to Memory Map Interface. Width must be equal or less than C_M_AXI_S2MM_DATA_WIDTH .	C_S_AXIS_S2MM_TDATA_ WIDTH	32,64,128, 256,512, 1024	32	integer
Maximum burst size per burst request on Memory Map Write interface	C_S2MM_BURST_SIZE	16,32,64, 128,256	16	integer

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

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

These clocks can work in either synchronous mode (C_PRMRY_IS_ACLK_ASYNC=0) or asynchronous mode (C_PRMRY_IS_ACLK_ASYNC=1).

When asynchronous mode is 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`. In synchronous mode, all clocks must be at the same frequency and from the same source.

Resets

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

AXI DMA Simple DMA Operation

Simple DMA mode (C_INCLUDE_SG = 0) 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 assert 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 Simple DMA mode.
3. Write a valid source address to the MM2S_SA register. If the AXI DMA is not configured for Data Re-Alignment (C_INCLUDE_MM2S_DRE = 0 or C_M_AXIS_MM2S_TDATA_WIDTH > 64), then a valid address must be aligned or undefined results occur. What is considered aligned or unaligned is based on the stream data width C_M_AXIS_MM2S_TDATA_WIDTH.

   For example, if C_M_AXIS_MM2S_TDATA_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 C_M_AXIS_MM2S_TDATA_WIDTH = 64, data is aligned if it is located at double-word offsets (64-bit offsets), that is 0x0, 0x8, 0x10, 0x18, and so forth. If C_INCLUDE_MM2S_DRE = 1 and C_M_AXIS_MM2S_TDATA_WIDTH < 128, then 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.

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 Simple DMA mode.

3. Write a valid destination address to the S2MM_DA register. If the AXI DMA is not configured for Data Re-Alignment (C_INCLUDE_S2MM_DRE = 0 or C_S_AXIS_S2MM_TDATA_WIDTH > 64) then a valid address must be aligned or undefined results occur. What is considered aligned or unaligned is based on the stream data width C_S_AXIS_S2MM_TDATA_WIDTH.

   For example, if C_M_AXIS_MM2S_TDATA_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 C_M_AXIS_MM2S_TDATA_WIDTH = 64, data is aligned if it is located at double-word offsets (64-bit offsets), that is 0x0, 0x8, 0x10, 0x18, and so forth. If C_INCLUDE_S2MM_DRE = 1 and C_S_AXIS_S2MM_TDATA_WIDTH < 128 then Destination Addresses can be of any byte offset.
4. Write the length in bytes of the receive buffer in the S2MM_LENGTH register. A value of zero written 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 written that is less than the number of bytes received produces undefined results. The S2MM_LENGTH register must be written last. All other S2MM register can be written in any order.

Scatter Gather Descriptor (C_INCLUDE_SG = 1)

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 8 MB of data buffer per descriptor. Two descriptor chains are required for the two data transfer direction, MM2S and S2MM.

Note: Descriptors must be aligned on 16 32-bit word alignment. Example valid offsets are 0x00, 0x40, 0x80, 0xC0, and so forth.

Table 4-2: **Descriptor Fields**

Address Space Offset(1)	Name	Description
00h	NXTDESC	Next Descriptor Pointer
04h	RESERVED	N/A
08h	BUFFER_ADDRESS	Buffer Address
0Ch	RESERVED	N/A
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

Table 4-2: **Descriptor Fields**

Address Space Offset(1)	Name	Description
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, C_SG_INCLUDE_STSCNTRL_STRM = 1. When the Control/ Status Streams are not included (C_SG_INCLUDED_STSCNTRL_STRM = 0), the User Application fields are not fetched or updated by the Scatter Gather Engine.

MM2S_NXTDESC (MM2S Next Descriptor Pointer)

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

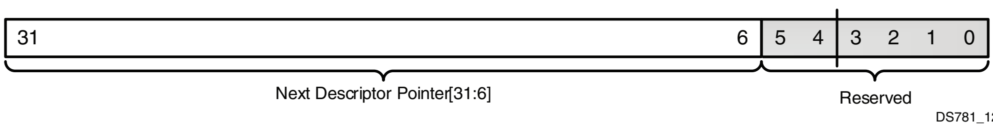


Figure 4-1: **MM2S_NXTDESC**

Table 4-3: **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_BUFFER_ADDRESS (MM2S Buffer Address)

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

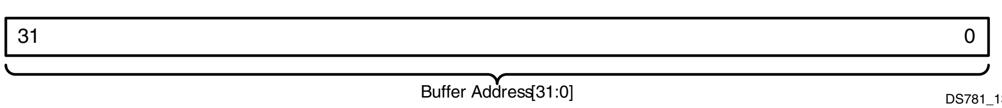


Figure 4-2: **MM2S Buffer Address**

Table 4-4: **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 (C_INCLUDE_MM2S_DRE = 1), 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 (C_INCLUDE_MM2S_DRE = 0), the Buffer Address must be MM2S stream data-width aligned.

MM2S_CONTROL (MM2S Control)

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

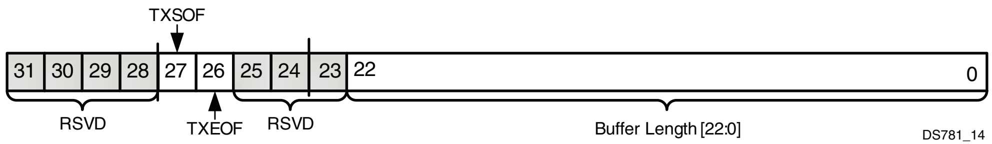


Figure 4-3: **MM2S_CONTROL**

Table 4-5: **MM2S_CONTROL Details**

Bits	Field Name	Description
22 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 C_SG_LENGTH_WIDTH. A maximum of 8 MB of transfer can be described by this field. ***Note:*** Setting C_SG_LENGTH_WIDTH smaller than 23 reduces FPGA resource utilization.
27 to 23	Reserved	These bits are reserved and should be set to zero.
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 S2MM (Receive), this bit is reserved and must be set to 0 by the CPU. For proper operation, there must be an SOF descriptor (TXSOF=1) and an 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:*** For C_SG_INCLUDE_STSCNTRL_STRM = 1, user application data from APP0 to APP4 of the 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.

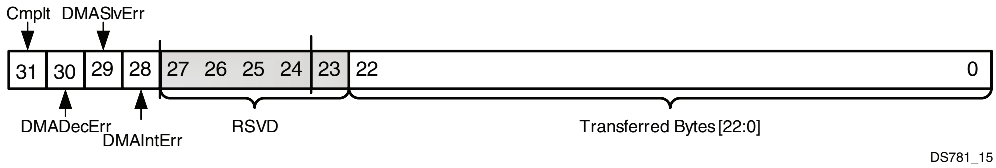


Figure 4-4: **MM2S_STATUS**

Table 4-6: **MM2S_STATUS Details**

Bits	Field Name	Description
22 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 C_SG_LENGTH_WIDTH. A maximum of 8 MB of transfer can be described by this field. ***Note:*** Setting C_SG_LENGTH_WIDTH smaller than 23 reduces FPGA resource utilization.
27 to 23	Reserved	These bits are reserved and should be set to zero.
28	DMAIntErr	DMA Internal Error. Internal Error detected by primary AXI DataMover. This error can occur if a 0 length bytes to transfer is fed to the AXI DataMover. 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 DataMover. 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 4-6: **MM2S_STATUS Details *(Cont'd)***

Bits	Field Name	Description
30	DMADecErr	DMA Decode Error. Decode Error detected by primary AXI DataMover. This error occurs if the address request is to an invalid address (that is, 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 when in Tail Pointer Mode (currently the only supported mode). • 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 (MM2S User Application Fields 0 to 4)

This value provides User Application fields for MM2S control stream.

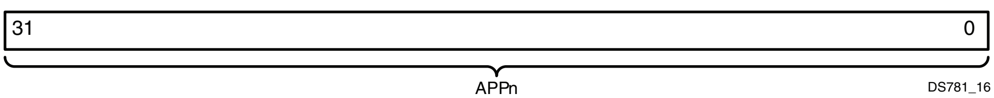


Figure 4-5: **MM2S_STATUS**

Table 4-7: **User Application Details**

Bits	Field Name	Description
31 to 0	APP0 to APP4	User application fields 0 to 4. Specifies user-specific application data. For C_SG_INCLUDE_STSCNTRL_STRM = 1, the Application (APP) fields of the SOF Descriptor are transmitted out the AXI Control Stream. For other MM2S descriptors with SOF = 0, the APP fields are fetched but ignored. ***Note:*** For C_SG_INCLUDE_STSCNTRL_STRM = 0, these fields are not fetched.

S2MM_NXTDESC (MM2S Next Descriptor Pointer)

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

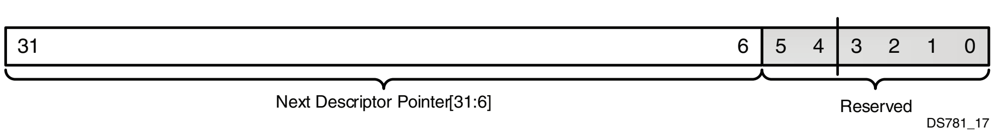


Figure 4-6: **S2MM_NXTDESC**

Table 4-8: **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_BUFFER_ADDRESS (S2MM Buffer Address)

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

Figure 4-7: **S2MM Buffer Address**

Table 4-9: **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 (C_INCLUDE_S2MM_DRE = 1), the Buffer Address can be at any byte offset. If Data Realignment Engine is not included (C_INCLUDE_S2MM_DRE = 0), the Buffer Address must be S2MM stream data width aligned.

S2MM_CONTROL (S2MM Control)

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

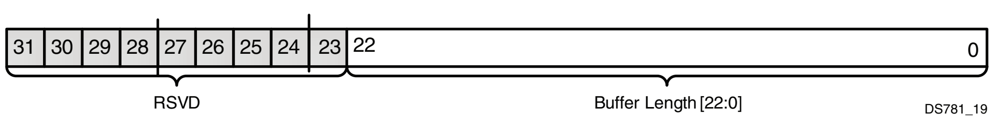


Figure 4-8: **S2MM_CONTROL**

Table 4-10: **S2MM_CONTROL Details**

Bits	Field Name	Description
22 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 C_SG_LENGTH_WIDTH. A maximum of 8 Mbytes of transfer can be described by this field. ***Note:*** The sum total of 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 C_SG_LENGTH_WIDTH smaller than 23 reduces FPGA resource utilization.
31 to 23	Reserved	These bits are reserved and should be set to zero.

S2MM_STATUS (S2MM Status)

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

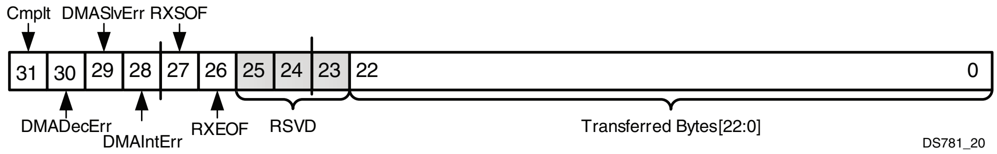


Figure 4-9: **S2MM_STATUS**

Table 4-11: **S2MM_STATUS Details**

Bits	Field Name	Description
22 to 0	Transferred Bytes	This value indicates the amount of data received and stored in the buffer described by this descriptor. This may or may not match the buffer length. For example, if this descriptor indicates a buffer length of 1024 bytes but only 50 bytes were received and stored in the buffer, then the Transferred Bytes field indicates 32h. The entire receive packet length can be determined by adding the Transferred Byte values from each descriptor from the RXSOF descriptor to the Receive Start Of Frame (RXEOF) descriptor. ***Note:*** The usable width of Transferred Bytes is specified by C_SG_LENGTH_WIDTH. A maximum of 8 Mbytes of transfer can be described by this field. ***Note:*** Setting C_SG_LENGTH_WIDTH smaller than 23 reduces FPGA resource utilization.
25 to 23	Reserved	These bits are reserved and should be set to zero.
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 CPU that the buffer associated with this descriptor contains the end of the packet. • 0 = Not end of frame. • 1 = End of frame. ***Note:*** For C_SG_INCLUDE_STSCNTRL_STRM = 1, User Application data sent via the status stream input is stored in APP0 to APP4 of the RXEOF descriptor.
27	RXSOF	Start of Frame. Flag indicating buffer holds first part of packet. This bit is set by AXI DMA to indicate to the CPU 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 DataMover. This error can occur if a 0 length bytes to transfer is fed to the AXI DataMover. 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 occurs, and C_SG_USE_STSAPP_LENGTH = 1, indicating less bytes or more bytes than what was actually commanded are received. 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.

Table 4-11: **S2MM_STATUS Details *(Cont'd)***

Bits	Field Name	Description
29	DMASlvErr	DMA Slave Error. Slave Error detected by primary AXI DataMover. 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 DataMover. This error occurs if the address request is to an invalid address (that is, 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 may 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.

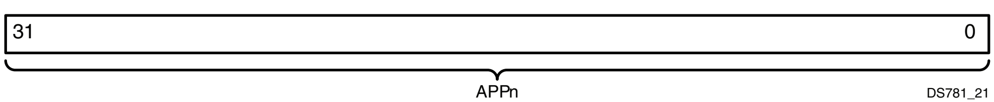


Figure 4-10: **S2MM_APP0 to S2MM_APP3**

Table 4-12: **User Application 0 to 3 Details**

Bits	Field Name	Description
31 to 0	APP0 to APP3	For C_SG_INCLUDE_STSCNTRL_STRM = 1, the status data received on the AXI Status Stream is stored into the APP fields of the EOF Descriptor. For other S2MM descriptors with EOF = 0, the APP fields are set to zero by the Scatter Gather Engine. ***Note:*** For C_SG_INCLUDE_STSCNTRL_STRM = 0, these fields are updated by the Scatter Gather Engine.

S2MM_APP4 (S2MM User Application Field 4)

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

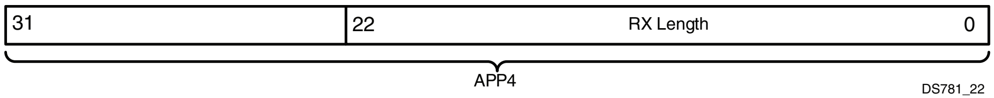


Figure 4-11: **S2MM_APP4**

Table 4-13: **User Application 4 Details**

Bits	Field Name	Description
31 to 0	APP4 / RxLength	User Application field 4 and Receive Byte Length. For C_SG_USE_STSAPP_LENGTH = 0, 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 EOF Descriptor. For C_SG_USE_STSAPP_LENGTH = 1, this field has a dual purpose. First, the least significant C_SG_LENGTH_WIDTH bits 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. ***Note:*** If using status application length (C_SG_USE_STSAPP_LENGTH=1), RxLength value must be stored in the C_SG_LENGTH_WIDTH least significant bits of the UserApp4 field in the AXI Status Stream Packet.

AXI DMA Scatter / Gather Operation

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. 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 utilized. 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's 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. Each descriptor contains, in the status field, the number of bytes actually transferred for that particular descriptor. The total number of bytes transferred for the receive packet can be determined by the software by walking from the RXSOF descriptor through the descriptor chain to the RXEOF descriptor.

To keep the Target IP on MM2S and/or S2MM AXI4-Streams synchronized with AXI DMA, reset out ports clocked in the primary clock domain (`axi_prmry_aclk`) are provided. These ports are `mm2s_prmry_reset_out`, `mm2s_cntrl_reset_out`, `s2mm_prmry_reset_out`, and `s2mm_sts_reset_out` and asserts low on assertion of `axi_resetn` and also on soft reset.

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 was 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 with the Complete bit set to 1.

While the software is processing descriptors, AXI DMA hardware is prevented from stepping on the descriptors being processed by software by the tail pointer. When the software finishes re-allocating a set of descriptors, it then moves the tail pointer location to the end of the re-allocated descriptors by writing to the associated channel's TAILDESC register. 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 simply continues to process descriptors until reaching the new tail descriptor pointer location.

The following illustrates descriptor processing by the AXI DMA engine utilizing tail pointer mode.

1. Initialization (DMACR.RS=0), as shown in Figure 4-12.
   - Descriptors are initialized in the remote memory by the software. The descriptors are arranged into a ring with each NXTDESC_PTR pointing to the next descriptor. Each descriptor points to a buffer for transmitting or receiving data.
   - The software then initializes the CURDESC_PTR register for the associated channel to point to BD1.

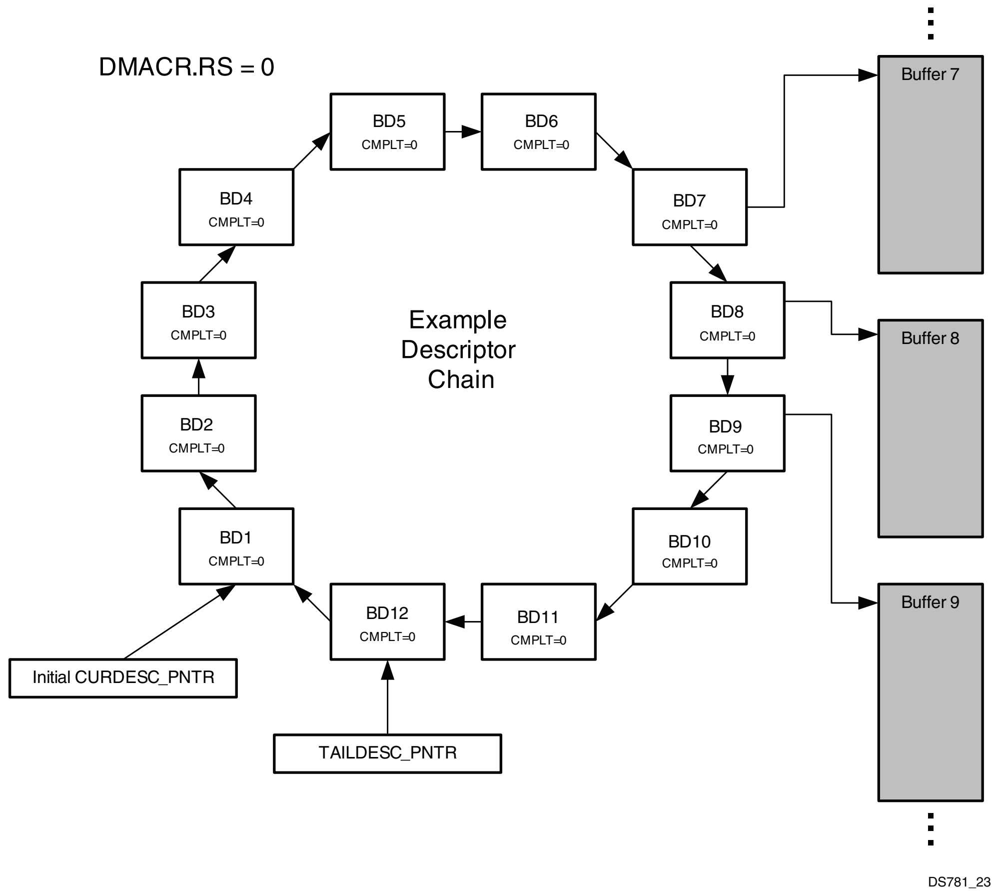


Figure 4-12: **Initialization**

2. Beginning execution (DMACR.RS=1), as shown in Figure 4-13.
   - The software sets the AXI DMA Engine to run (DMACR.RS=1).
   - The software then initializes the TAILDESC_PTR register for the associated channel to point to BD12.
   - AXI DMA fetches descriptors until the internal scatter gather fetch queue for the associated channel is full, as indicated by the shaded boxes.
   - AXI DMA updates the CURDESC_PTR in the Scatter Gather Engine with the fetched NXTDESC_PTR to correctly fetch the next descriptor.
   - AXI DMA pulls descriptors from the fetch queue to begin actual processing of the transfer described by the descriptor. The CURDESC_PTR register then updates reflecting the actual descriptor being processed by the engine. This can be read by the software, but it should be noted that this value changes while DMASR.Halted = 0.

   Note: On error, the DMA engine halts and the CURDESC_PTR register is updated to the descriptor associated with the error.

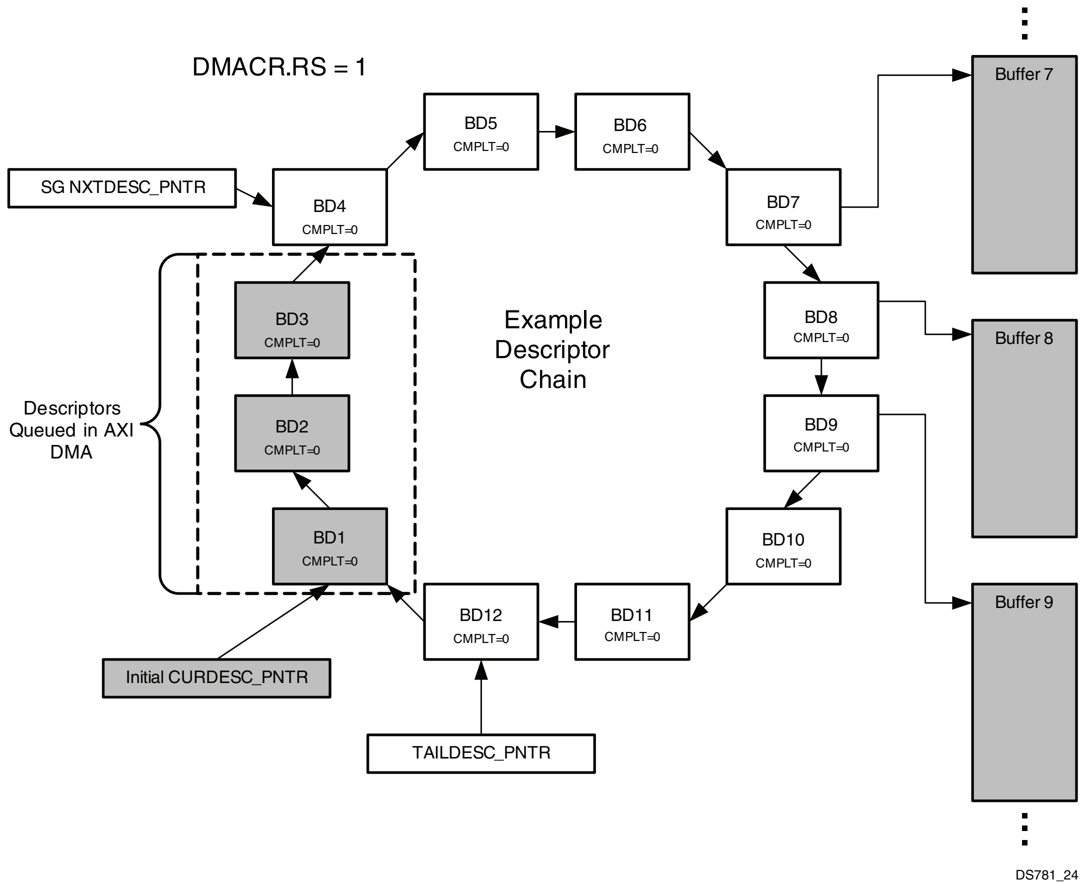


Figure 4-13: **Beginning Execution**

3. Continued execution (DMACR.RS=1), as shown in Figure 4-14.
   - As long as the CURDESC_PTR does not equal the TAILDESC_PTR and there is room for a fetched descriptor to be stored in the fetch queue, descriptors continue to be fetched by the Scatter Gather Engine.
   - As descriptors are processed by the DMA Controller, they are updated to the Scatter Gather engine update queue. When the AXI DataMover status is available for the associated transfer, the descriptor is updated to remote memory.
   - Descriptors processed by the AXI DMA have the Completed bit set in the STS/Control word of the descriptor. These descriptors are free for the software to re-allocate.

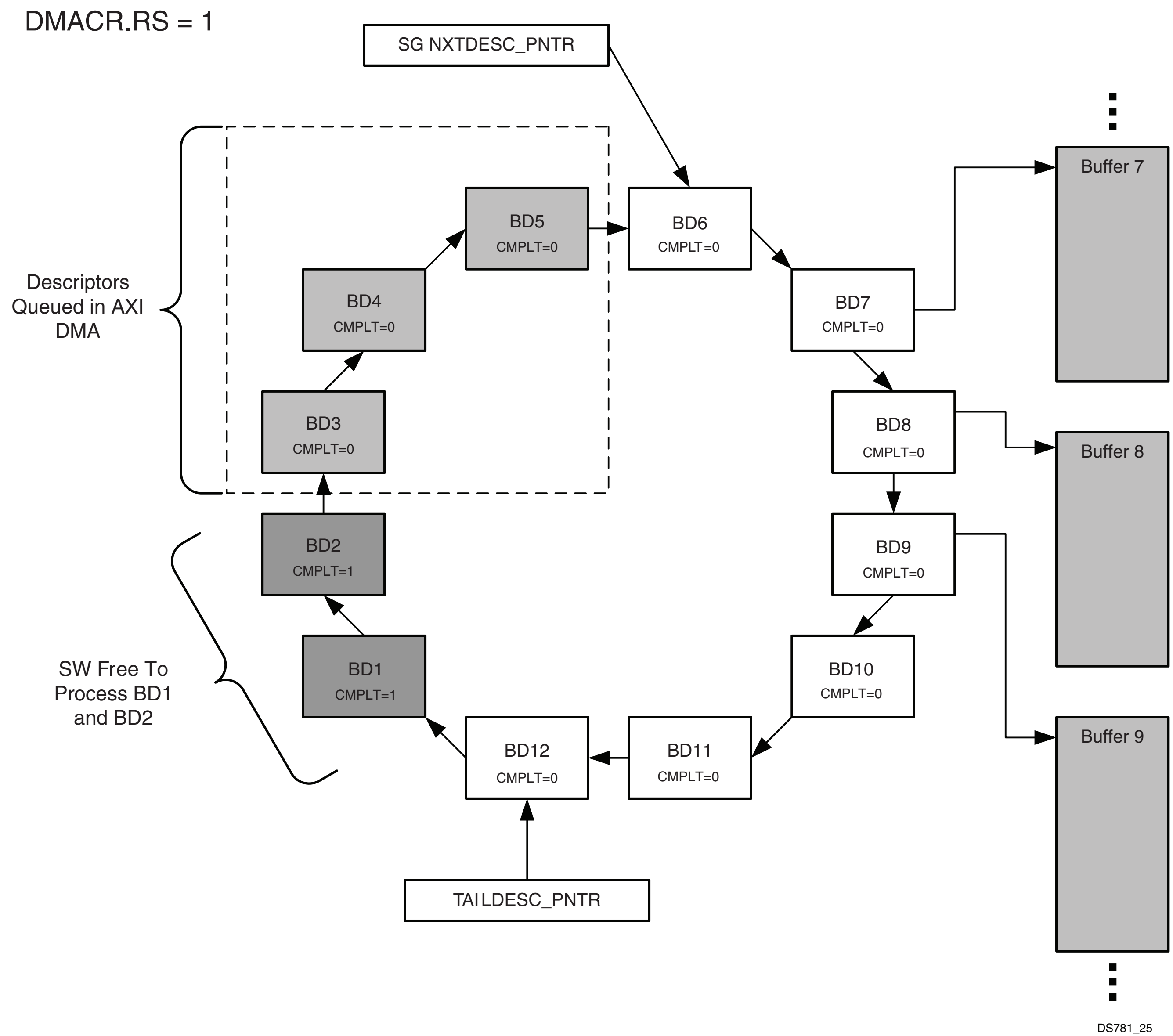


Figure 4-14: **Continued Execution**

4. Descriptors re-allocated (DMACR.RS=1), as shown in Figure 4-15.
   - After the software has processed data on the completed descriptors, it can re-allocate the descriptors by setting the complete bit to 0 and adjusting any other control bits, buffer pointer, buffer length, and others.
   - After re-allocating the descriptors, the software then moves the TAILDESC_PTR to point to the last re-allocated descriptor.
   - If AXI DMA was paused (DMASR.Idle=1) because it had hit the initial TAILDESC_PTR location, it automatically re-starts descriptor fetching when the software writes the new TAILDESC_PTR.

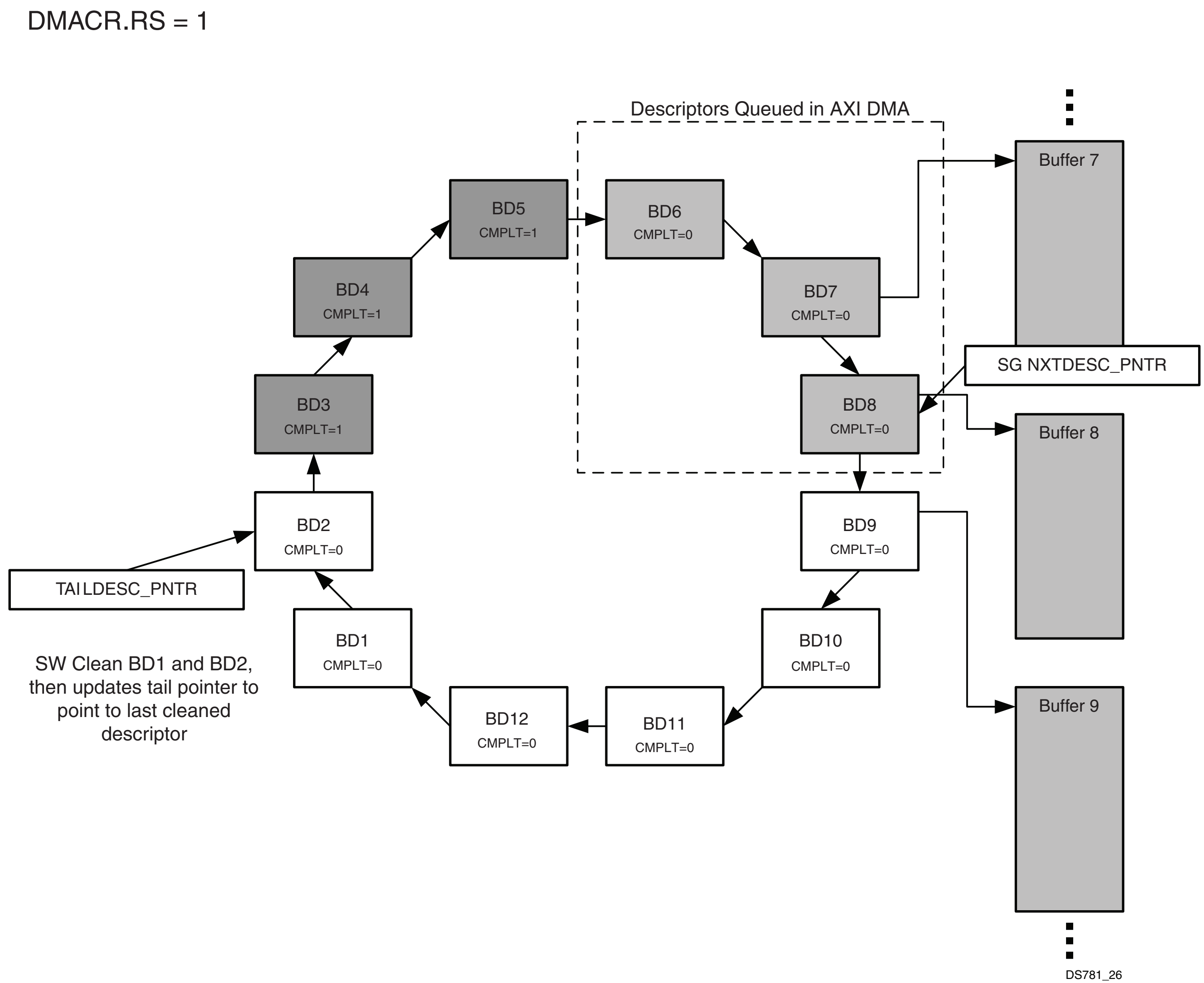


Figure 4-15: **Descriptors Re-Allocated**

MM2S Descriptor Settings/Relationship

The relationship between descriptor SOF/EOF settings and the AXI Control Stream are illustrated in Figure 4-16. 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 AXI Control Stream is included (C_SG_INCLUDE_STSCNTRL_STRM = 1). User Application fields following descriptor with SOF=1, up to and including descriptor with EOF =1, are ignored by the AXI DMA engine. If C_SG_INCLUDE_STSCNTRL_STRM = 0, the User Application fields are not fetched by the SG Fetch Engine.

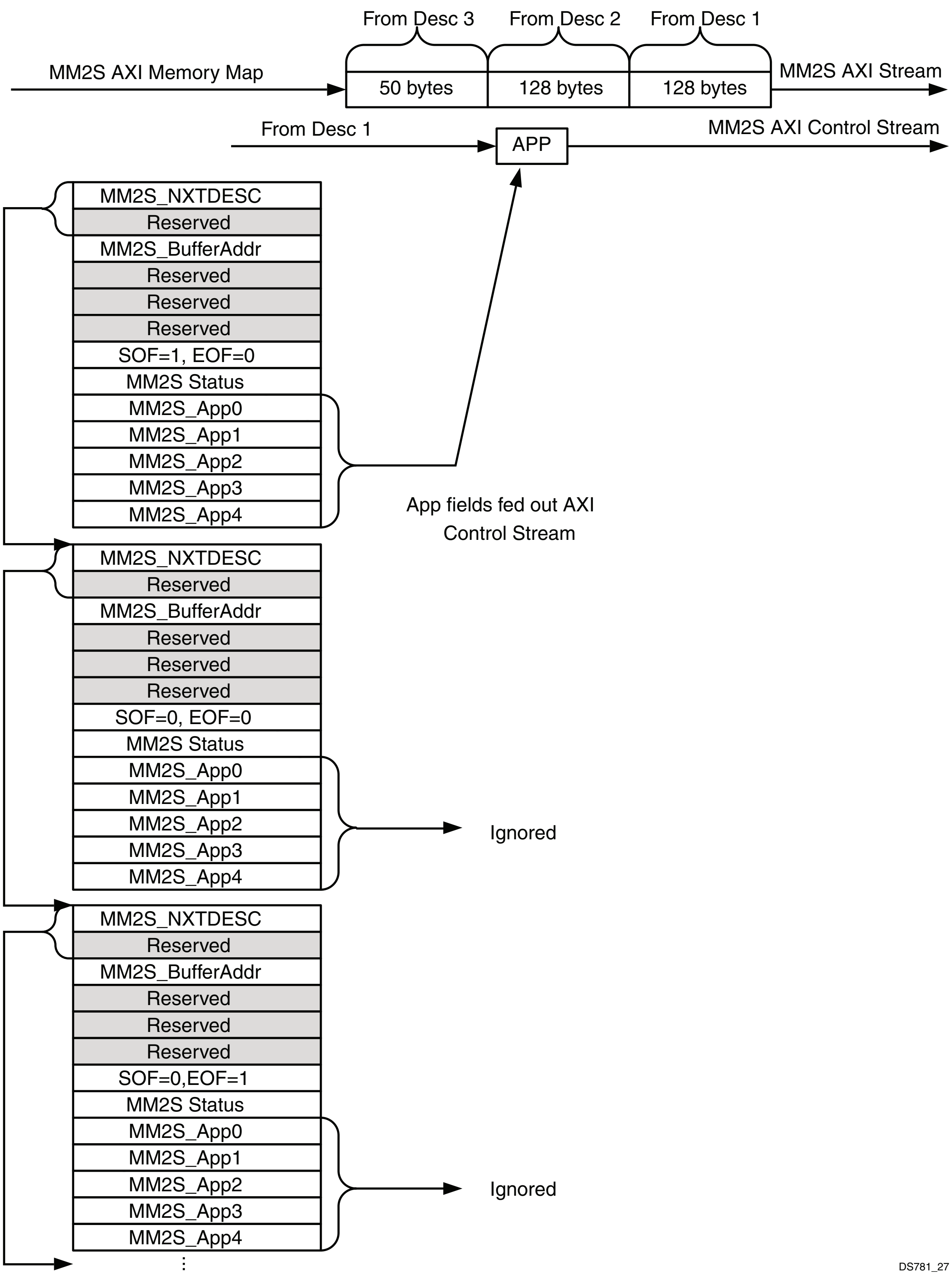


Figure 4-16: **Detail of Descriptor Relationship to MM2S Stream and Control Stream**

Control Stream Detail

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. Figure 4-17 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.

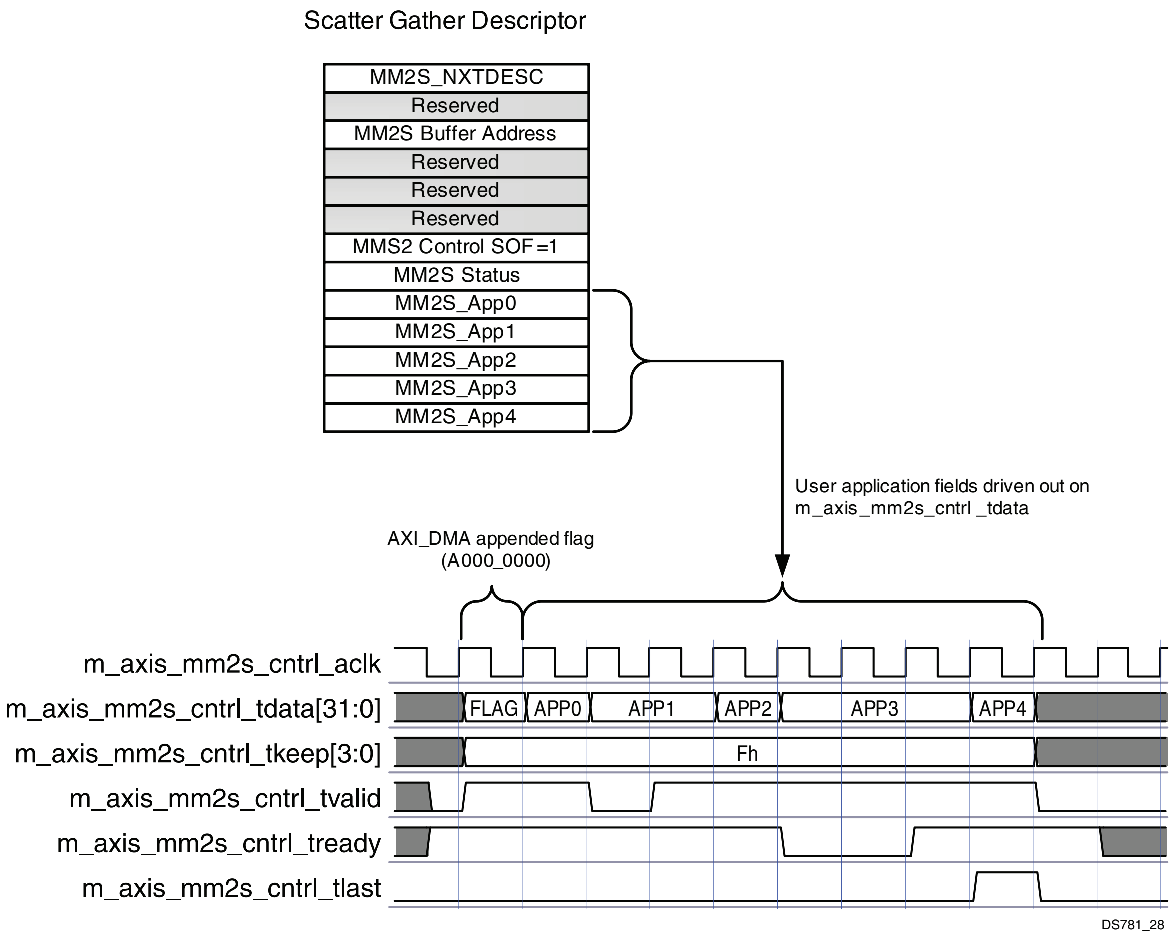


Figure 4-17: **Example User Application Field / Timing for MM2S Control Stream**

S2MM Descriptor Settings/Relationship

The relationship between descriptor RXSOF/RXEOF settings and the AXI Status Stream are illustrated in Figure 4-18. 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 (that is, the sum total buffer lengths in each descriptor of the descriptor chain) to be greater than or equal to the maximum sized packet that is received.

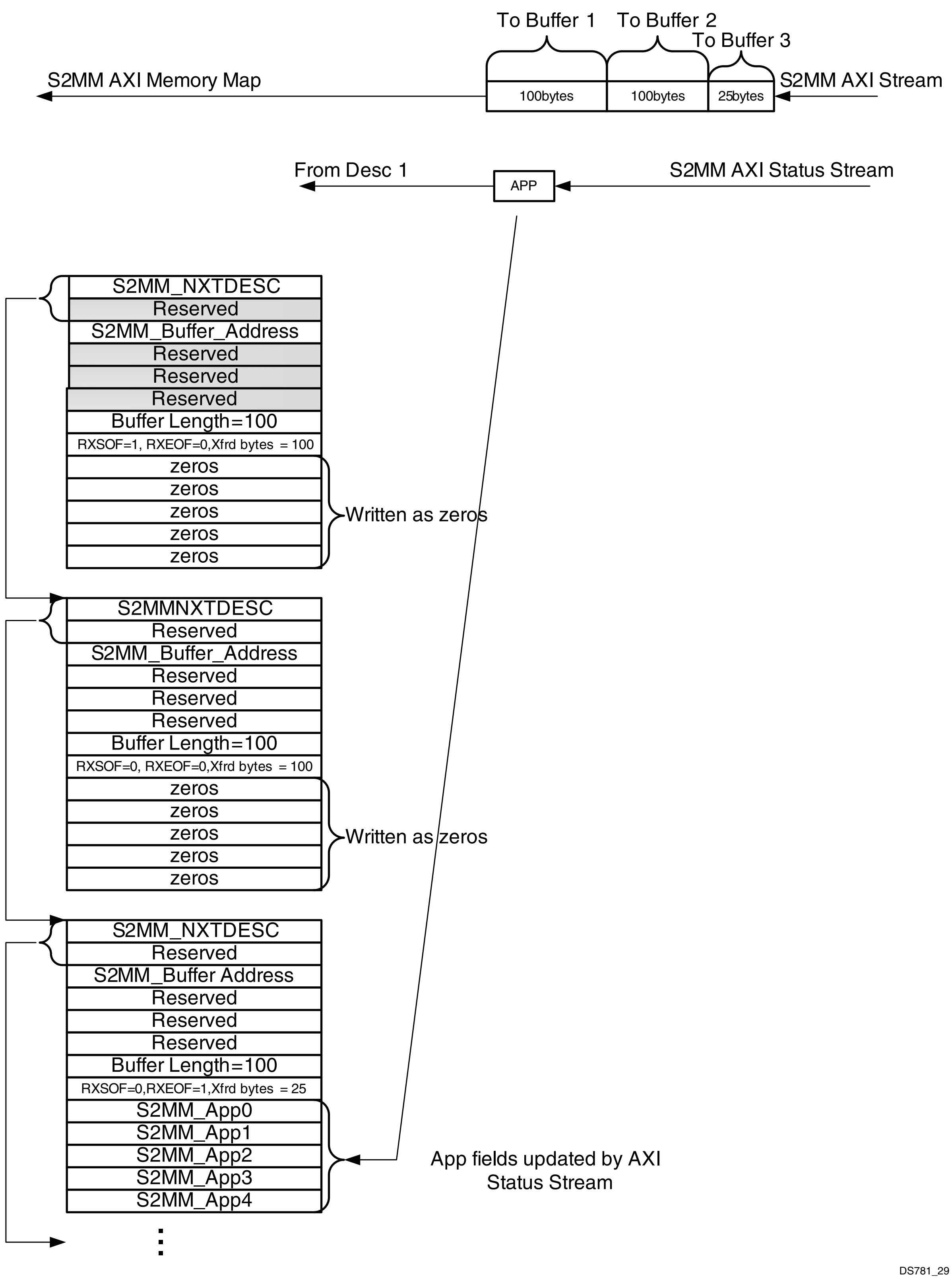


Figure 4-18: **Detail of Descriptor Relationship to S2MM Stream and Status Stream**

If the AXI Status Stream is included (C_SG_INCLUDE_STSCNTRL_STRM = 1), 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 a user provides the total length in the status stream, this value is stored in the user-defined application location in the descriptor with RXEOF=1.

Status Stream Detail

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 Figure 4-19, the status packet updates to the app fields of the detected last descriptor (RXEOF = 1) describing the packet.

S2MM Throttling

Throttling by the source device is allowed on the AXI Status Stream. Throttling by AXI DMA occurs on the AXI Status Stream if the status FIFO fills up.

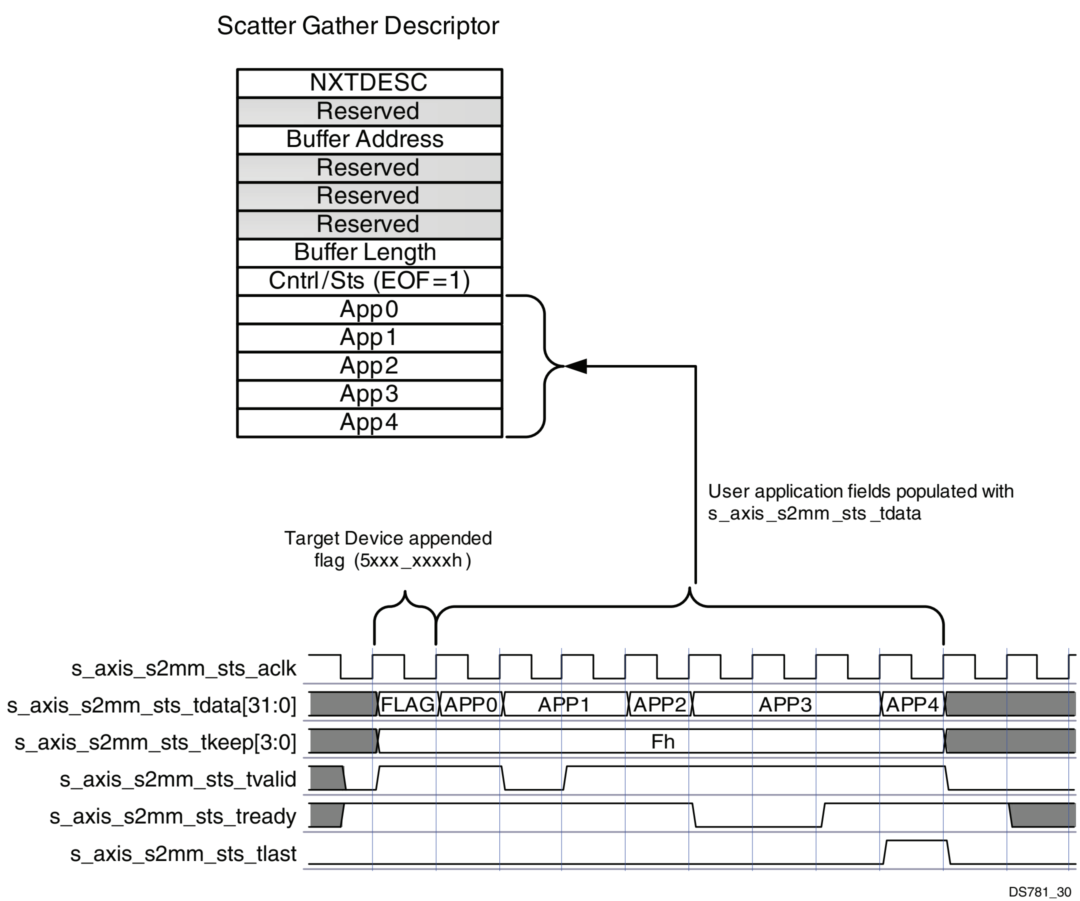


Figure 4-19: **Example User Application Field / Timing for S2MM Status Stream**

AXI Status Stream Receive Length Timing

If the AXI DMA is enabled to use a receive length (C_SG_USE_STSAPP_LENGTH = 1) received via AXI Status Stream, this is used for S2MM command calculation. Example timing is illustrated in Figure 4-20.

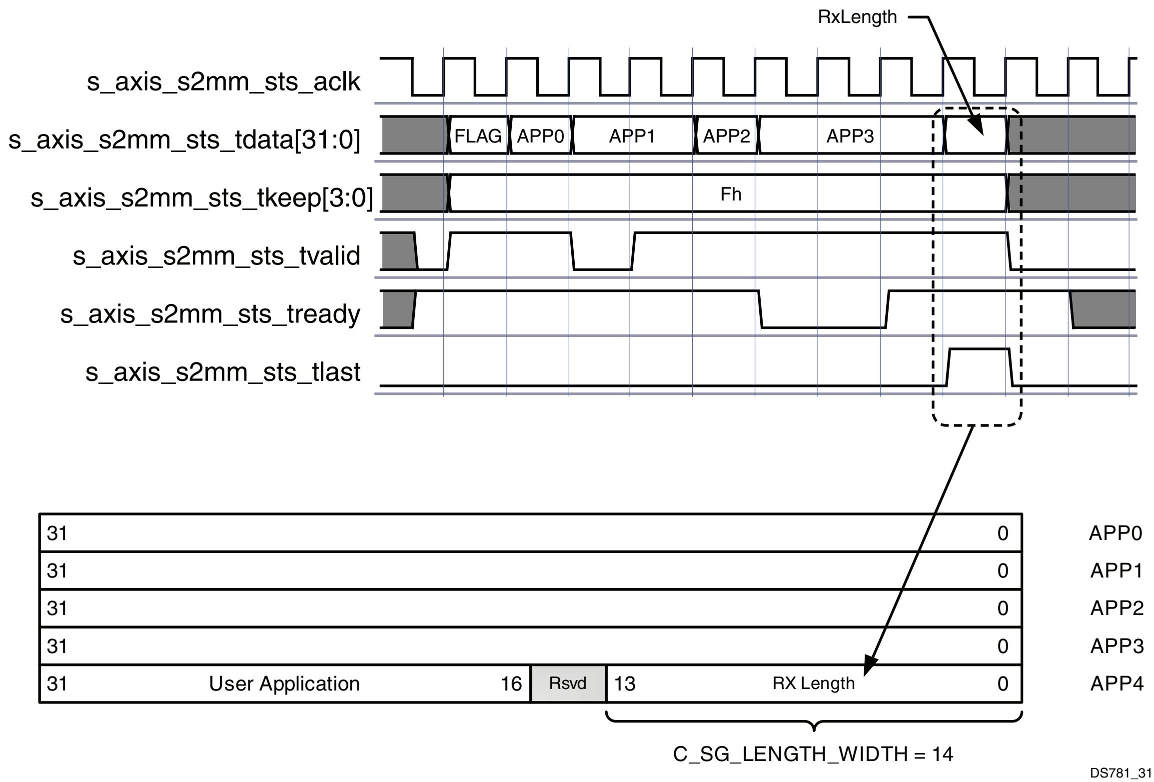


Figure 4-20: **Receive Length Timing**

AXI DMA System Configuration

System configuration should be taken into consideration when using AXI DMA. Due to the high performance and low latency design of AXI DMA, throttling or back pressure on the AXI4-Stream ports of AXI DMA (MM2S and S2MM) subsequently applies back pressure on the associated channel's AXI4 Memory Map side. Depending on a user's system configuration this back pressure can lead to a deadlock situation where, for example, a write transfer on S2MM to a single ported memory controller cannot complete because of a throttled read transfer on MM2S. A loopback type system where the MM2S stream interface is looped back to S2MM stream interface, such as when loopback is turned on in AXI Ethernet, can present such a deadlock scenario.

The AXI Interconnect interfacing to the AXI DMA AXI4 Memory Map ports on MM2S and S2MM can be configured to prevent the deadlock scenario described previously. Read and Write Data FIFOs can be turned on in the AXI Interconnect to allow read and/or write data to be buffered up. Enabling the FIFOs along with limiting the number of outstanding read and write requests accepted by AXI Interconnect guarantees that all requested data to be transferred can be accepted by the AXI Interconnect preventing deadlock.

To enable these AXI Interconnect features, four non-hdl parameters are provided:

- C_INTERCONNECT_M_AXI_MM29
- C_INTERCONNECT_M_AXI_MM2S_READ_ISSUING
- C_INTERCONNECT_M_AXI_MM2S_READ_FIFO_DEPTH
- C_INTERCONNECT_M_AXI_S2MM_WRITE_ISSUING
- C_INTERCONNECT_M_AXI_S2MM_WRITE_FIFO_DEPTH

Setting these parameters correctly configures the AXI Interconnect interfaced to the Memory Map Ports of the AXI DMA. For AXI DMA *_ISSUING multiplied by the associated channels *_BURST_SIZE must be less than the *_FIFO_DEPTH to prevent the deadlock scenario. For example, if C_MM2S_BURST_SIZE is set to 16 and C_INTERCONNECT_M_AXI_MM2S_READ_ISSUING is set to 4, then the product of these two values would be 16 X 4 = 64.

Therefore the setting C_INTERCONNECT_M_AXI_MM2S_READ_FIFO_DEPTH = 512 would be sufficient to satisfy the requirements (16 X 4 = 64 which is less than 512). Here is the formula presented for clarity:

*_BURST_SIZE X *_ISSUING < *_FIFO_DEPTH

If building AXI DMA using the EDK tool suite, the issuing parameter is set automatically based on the AXI DMA burst size parameter and the fifo depth parameter is set to 512. When looking at FPGA resource utilization in an EDK system with AXI DMA, the user should note that the AXI Interconnect instantiates FIFOs for both MM2S and S2MM channels of AXI DMA.

Interrupt Controller

An interrupt output is provided for each channel (MM2S and S2MM). For Scatter / Gather mode (C_INCLUDE_SG = 1) this output then drives high when the interrupt threshold is met, if there is a delay interrupt, or on error if the associated interrupt is enabled, as shown in Figure 4-21.

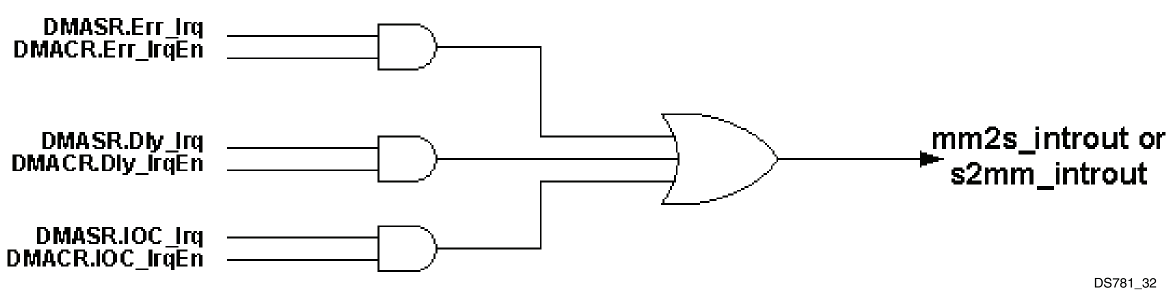


Figure 4-21: **Interrupt Out Concept**

For Simple DMA mode (C_INCLUDE_SG = 0) the interrupt output will drive high at the completion of a transfer if the DMACR.IOC_IrqEn for the associated channel is set.

Threshold Interrupt (Scatter / Gather Only)

Interrupt coalescing can be accomplished by setting the DMACR.IRQThreshold field. With each Interrupt On Complete event (transmitted end of packet [descriptor with TXEOF=1] or received end of packet), the threshold count is decremented. When the count reaches zero, an `IOC_Irq` is generated. If `IOC_IrqEn` = 1, an interrupt is generated on the associated channels introut signal (`mm2s_introut` or `s2mm_introut`). When a threshold interrupt is generated, the threshold count is reloaded in the threshold counter in preparation for the next threshold event. The internal threshold count value is presented to software in DMASR.IRQThresholdSts. A DMACR.IRQThreshold value of 0x01 (default) causes a single IOC interrupt event to immediately generate an interrupt out.

If the delay interrupt is enabled (DMACR.IRQDelay not equal to 0), a delay interrupt event also reloads the internal threshold counter. Finally, with each software write of the threshold value (DMACR.Threshold), the internal threshold counter is reloaded.

Figure 4-22 illustrates the functional composition of the interrupt threshold logic.

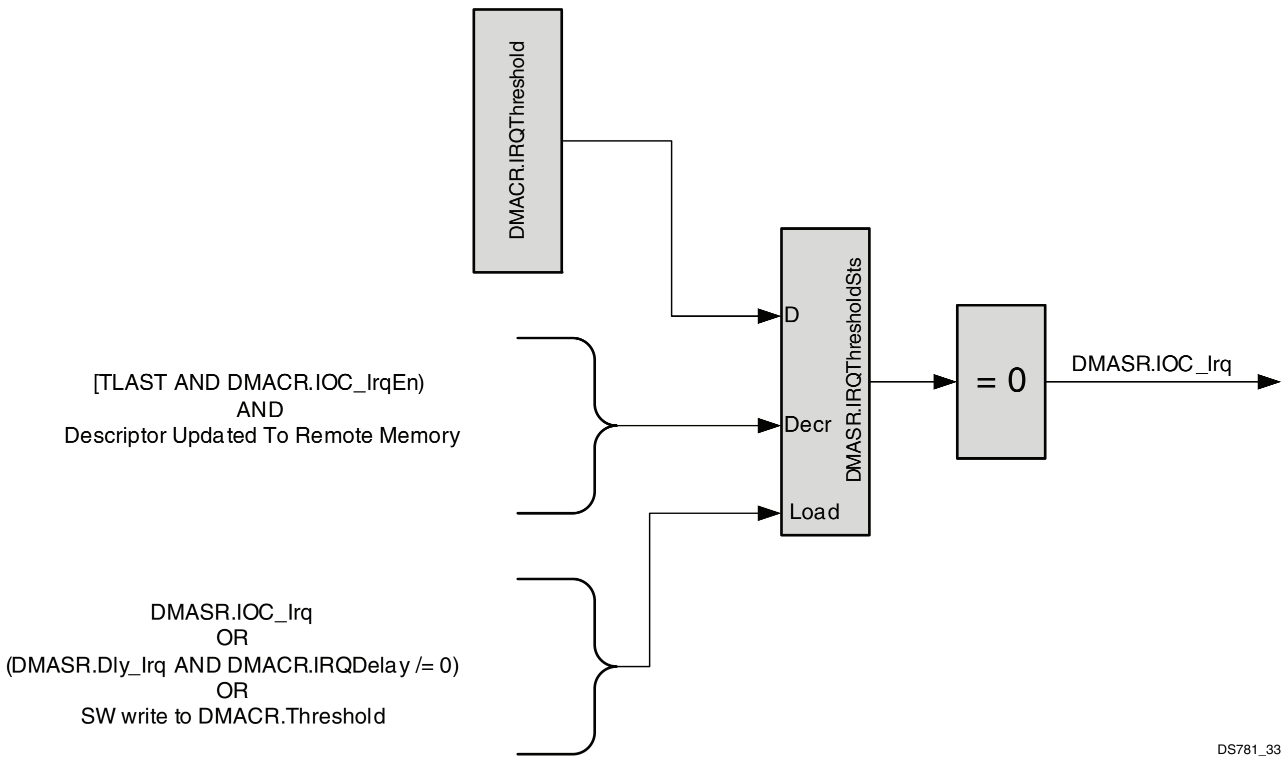


Figure 4-22: **Interrupt On Complete Threshold Logic Concept**

Delay Interrupt (Scatter / Gather Only)

The delay interrupt is a mechanism by which software can receive an interrupt even when the interrupt threshold is not met. This is useful primarily for the S2MM (receive) channel for when receive data is sporadic. The software can still get an interrupt and service what data is received even if the threshold count has not been decremented to zero. Figure 4-23 shows a high-level block diagram of the delay interrupt architecture.

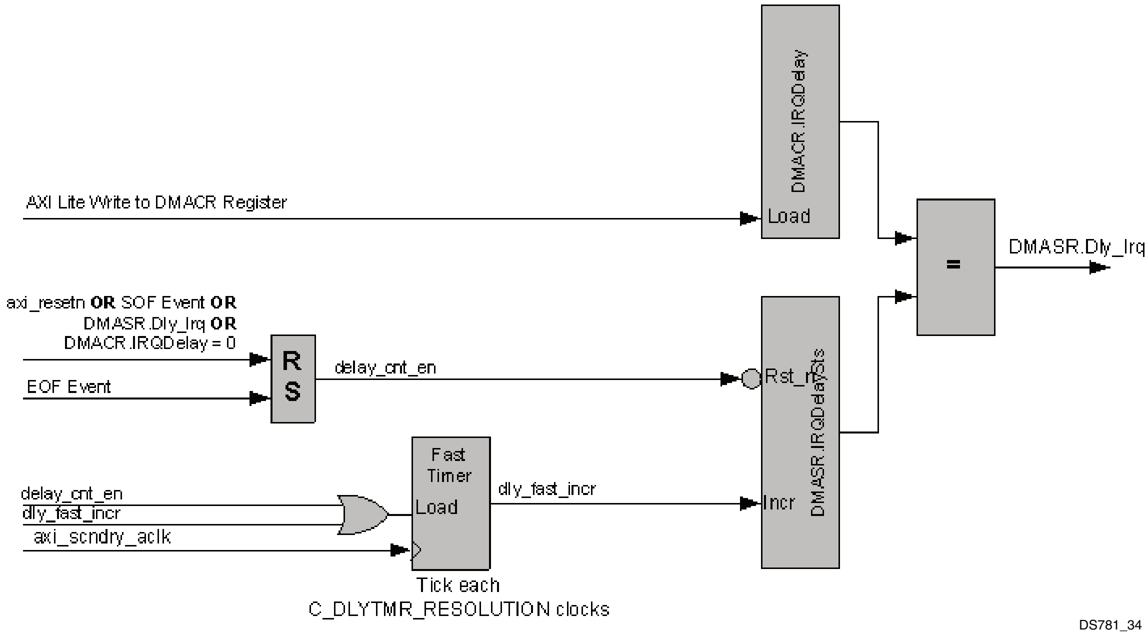


Figure 4-23: **Delay Interrupt Logic Concept**

The delay interrupt timer is enabled by setting DMACR.IRQDelay value to a non-zero value. The delay time begins counting up upon receipt of the end of packet (EOF Event), as indicated by descriptor with EOF=1. The delay time is reset with each subsequent start of packet (SOF Event), as indicated by descriptor with SOF=1. When a delay interrupt event occurs, the delay timer is reset to zero (DMASR.IRQDelaySts=0x00), and the delay timer does not count until the CPU services the interrupt by clearing the DMASR.Dly_Irq bit to 0. Figure 4-24 shows example timing for this situation.

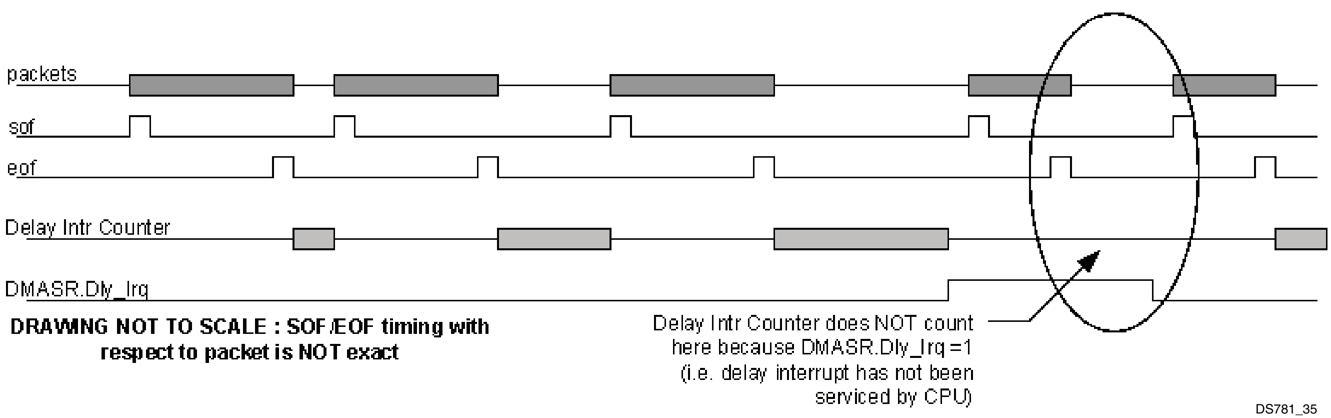


Figure 4-24: **Example Delay Timer Timing**

Errors

Any detected error results in the AXI DMA gracefully halting. When an error is detected, the errored channel's DMACR.RS bit is set to 0. Per AXI protocol, all AXI transfers must complete. Therefore, the AXI DMA completes all pending transactions before setting the errored channel's DMASR.Halted bit. When the DMASR.Halted bit is set to 1, the AXI DMA engine is truly halted.

For Scatter / Gather Mode the pointer to the descriptor associated with the errored transfer is updated to the channels CURDESC_PTR register. On the rare occurrence that more than one error is detected, only the CURDESC_PTR register for one of the errors is logged. To resume operations, a reset must be issued to the AXI DMA.

The following is a list of possible errors:

- **DMAIntErr**. DMA Internal Error indicates that an internal error in the AXI DataMover was detected. For Scatter / Gather Mode (C_INCLUDE_SG = 1) this can occur under two conditions. First, for MM2S and S2MM channels, it can occur when a BTT = 0 is written to the primary AXI DataMover. This would happen if a descriptor is fetched with the buffer length = 0.

  Secondly, for S2MM channels, the internal error can occur if C_SG_USE_STSAPP_LENGTH = 1 and an underflow condition occurs on the S2MM primary stream interface. This indicates that a failure in the system has occurred. For example, a soft reset issued in the stream source interrupts the receive stream, or a reported length on the status stream does not match actual data received.

  For Simple DMA Mode (C_INCLUDE_SG = 0) this error cannot occur.
- **DMASlvErr**. DMA Slave Error occurs when the slave to or from which data is transferred responds with a SLVERR.
- **DMADecErr**. DMA Decode Error occurs when the address request is targeted to an address that does not exist.
- **SGIntErr**. Scatter Gather Internal Error occurs when a BTT = 0. This error only occurs if a fetched descriptor already has the Complete bit set. This condition indicates to the AXI DMA that the descriptor has not been processed by the CPU, and therefore is considered stale. This error is for Scatter / Gather Mode only (C_INCLUDE_SG = 1).
- **SGSlvErr**. Scatter Gather Slave Error occurs when the slave to or from which descriptors are fetched and updated responds with a SLVERR. This error is for Scatter / Gather Mode only (C_INCLUDE_SG = 1).
- **SGDecErr**. Scatter Gather Decode Error occurs when the address request is targeted to an address that does not exist. This error is for Scatter / Gather Mode only (C_INCLUDE_SG = 1).

Note: Scatter Gather error bits are unable to be updated to the descriptor in remote memory. They are only captured in the associated channel DMASR where the error occurred.

Error Priority

Table 4-14 shows the error priorities and when the errors might occur.

Table 4-14: **Error Priority**

Priority (1= highest priority)	Error	Occurrence
1	Fetch SGSlvErr	Cannot occur simultaneous with Fetch SGDecErr. Can occur simultaneous with Fetch SGIntErr (Stale Descriptor Error), and depending on timing can occur with one of the Update SG errors (Update SGSlvErr or Update SGDecErr).
1	Fetch SGDecErr	Cannot occur simultaneous with Fetch SGSlvErr. Can occur simultaneous with Fetch SGIntErr (Stale Descriptor Error), and depending on timing can occur with one of the Update SG errors (Update SGSlvErr or Update SGDecErr).
1	Fetch SGIntErr (Stale Data Error)	Can occur simultaneous with one of the SG fetch errors (Fetch SGDecErr or Fetch SGSlvErr), and depending on timing can occur with one of the Update SG errors (Update SGSlvErr or Update SGDecErr).
1	Update SGSlvErr	Cannot occur simultaneous with Update SGDecErr. Depending on timing, error can occur simultaneous with Fetch SGIntErr (Stale Descriptor Error) and one of the fetch SG errors (Fetch SGSlvErr or Fetch SGDecErr).
1	Update SGDecErr	Cannot occur simultaneous with Update SGSlvErr. Depending on timing, error can occur simultaneous with Fetch SGIntErr (Stale Descriptor Error) and one of the fetch SG errors (Fetch SGSlvErr or Fetch SGDecErr).
2	DMAIntErr (Zero Length Error)	Cannot occur simultaneous with DMASlvErr or DMADecErr for the descriptor with the zero length error. Cannot occur simultaneously with Fetch SGDecErr, Fetch SGSlvErr, or Fetch SGIntErr for the descriptor with the zero length error. Can occur simultaneously with errors detected for other descriptors not having zero length error.
3	DMAIntErr (Overflow/Underflow)	Cannot occur simultaneous with zero length error. Cannot occur simultaneously with Fetch SGDecErr, or Fetch SGSlvErr for the descriptor associated with overflow or underflow. Can occur simultaneous with DMASlvErr or DMADecErr.
3	DMADecErr	Cannot occur simultaneous with DMASlvErr.
3	DMASlvErr	Cannot occur simultaneous with DMADecErr.

Interconnect Parameters

The AXI DMA Interconnect Parameters are described in Table 4-15.

Table 4-15: **AXI DMA Interconnect Parameters**

Non-HDL Parameter	Allowable Values	Description
C_INTERCONNECT_M_AXI_MM2S _READ_ISSUING	1, 2, 4	This parameter sets the number of outstanding read requests the AXI Interconnect accepts from the AXI DMA MM2S Read Master. This parameter configures the AXI Interconnect slave port connected to the MM2S Read Master of AXI DMA and is set automatically by the EDK Tool Suite.
C_INTERCONNECT_M_AXI_S2MM _WRITE_ISSUING	1, 2, 4	This parameter sets the number of outstanding write requests the AXI Interconnect accepts from the AXI DMA S2MM Write Master. This parameter configures the AXI Interconnect slave port connected to the S2MM Write Master of AXI DMA and is set automatically by the EDK Tool Suite.
C_INTERCONNECT_M_AXI_MM2S _READ_FIFO_DEPTH	0, 32, 512	This parameters sets the read data FIFO depth (in elements) for AXI DMA MM2S Read Master. This parameter configures the AXI Interconnect slave port connected to the MM2S Read Master of AXI DMA and is set automatically by the EDK Tool Suite.
C_INTERCONNECT_M_AXI_S2MM _WRITE_FIFO_DEPTH	0, 32, 512	This parameters sets the write data First In First Out (FIFO) depth (in elements) for AXI DMA S2MM Write Master. This parameter configures the AXI Interconnect slave port connected to the S2MM Write Master of AXI DMA and is set automatically by the EDK Tool Suite.

Allowable Parameter Combinations

The AXI DMA Allowable Parameters Combinations are described in Table 4-16.

Table 4-16: **Allowable Parameter Combination**

Parameter Name	Affects Parameter	Relationship Description
C_INCLUDE_MM2S	C_INCLUDE_MM2S_DRE C_MM2S_BURST_SIZE	Affected Parameters are ignored when C_INCLUDE_MM2S = 0
C_INCLUDE_S2MM	C_INCLUDE_S2MM_DRE C_S2MM_BURST_SIZE	Affected Parameters are ignored when C_INCLUDE_S2MM = 0
C_INCLUDE_SG	C_SG_USE_STSAPP_LENGTH C_SG_INCLUDE_STSCNTRL_ STRM	Affected Parameters are ignored when C_INCLUDE_SG = 0
C_SG_INCLUDE_STSCNTRL_STRM	C_SG_USE_STSAPP_LENGTH	Affected Parameter is ignored when C_SG_INCLUDE_STSCNTRL_STRM = 0
C_M_AXI_MM2S_DATA_WIDTH	C_M_AXIS_MM2S_TDATA_ WIDTH	Affected Parameter must be less than or equal to C_M_AXI_MM2S_DATA_WIDTH
C_M_AXI_S2MM_DATA_WIDTH	C_S_AXIS_S2MM_TDATA_ WIDTH	Affected Parameter must be less than or equal to C_M_AXI_S2MM_DATA_WIDTH

Parameter - I/O Signal Dependencies

The AXI DMA I/O Signal Dependencies are described in Table 4-17.

Table 4-17: **Parameter - I/O Signal Dependencies**

Parameter Name	Affects Signal	Depends on Parameter	Relationship Description
C_M_AXI_SG_DATA_WIDTH	m_axi_sg_wdata, m_axi_sg_wstrb, m_axi_sg_rdata		The setting of the parameter sets the vector width of the port.
C_M_AXI_SG_ADDR_WIDTH	m_axi_sg_awaddr, m_axi_sg_araddr		The setting of the parameter sets the vector width of the port.
C_SG_INCLUDE_STSCNTRL_STRM	m_axis_mm2s_cntrl_tdata, m_axis_mm2s_cntrl_tkeep, m_axis_mm2s_cntrl_tvalid, m_axis_mm2s_tready, m_axis_mm2s_cntrl_tlast, mm2s_cntrl_reset_out_n, s_axis_s2mm_sts_tdata, s_axis_s2mm_sts_tkeep, s_axis_s2mm_sts_tready, s_axis_s2mm_sts_tvalid, s_axis_s2mm_sts_tlast, s2mm_sts_reset_out_n		If the parameter is assigned a value of zero, the output ports are tied to 0, and the input ports are left open.

Table 4-17: **Parameter - I/O Signal Dependencies *(Cont'd)***

Parameter Name	Affects Signal	Depends on Parameter	Relationship Description
C_INCLUDE_SG	m_axi_sg_aclk, m_axis_sg_araddr, m_axi_sg_arlen, m_axi_sg_arsize, m_axi_sg_arburst, m_axi_sg_arprot, m_axi_sg_arcache, m_axi_sg_arvalid, m_axi_sg_arready, m_axi_sg_rdata, m_axi_sg_rresp, m_axi_sg_rlast, m_axi_sg_rvalid, m_axi_sg_rready, m_axi_sg_awaddr, m_axi_sg_awlen, m_axi_sg_awsize, m_axi_sg_awburst, m_axi_sg_awprot, m_axi_sg_awcache, m_axi_sg_awvalid, m_axi_sg_awready, m_axi_sg_wdata, m_axi_sg_wstrb, m_axi_sg_wlast, m_axi_sg_wvalid, m_axi_sg_wready, m_axi_sg_bresp, m_axi_sg_bvalid, m_axi_sg_bready, m_axis_mm2s_cntrl_tdata, m_axis_mm2s_cntrl_tkeep, m_axis_mm2s_cntrl_tvalid, m_axis_mm2s_tready, m_axis_mm2s_cntrl_tlast, mm2s_cntrl_reset_out_n, s_axis_s2mm_sts_tdata, s_axis_s2mm_sts_tkeep, s_axis_s2mm_sts_tready, s_axis_s2mm_sts_tvalid, s_axis_s2mm_sts_tlast, s2mm_sts_reset_out_n		If the parameter is assigned a value of zero, the output ports are tied to 0 and the input ports are left open.

Table 4-17: **Parameter - I/O Signal Dependencies *(Cont'd)***

Parameter Name	Affects Signal	Depends on Parameter	Relationship Description
C_M_AXIS_MM2S_ CNTRL_TDATA_WIDTH	m_axis_mm2s_cntrl_tdata, m_axis_mm2s_cntrl_tkeep		The setting of the parameter sets the vector width of the port.
C_S_AXIS_S2MM_STS_ TDATA_WIDTH	s_axis_s2mm_sts_tdata, s_axis_s2mm_sts_tkeep		The setting of the parameter sets the vector width of the port.
C_INCLUDE_MM2S	m_axis_mm2s_araddr, m_axi_mm2s_arlen, m_axi_mm2s_arsize, m_axi_mm2s_arburst, m_axi_mm2s_arprot, m_axi_mm2s_arcache, m_axi_mm2s_arvalid, m_axi_mm2s_arready, m_axi_mm2s_rdata, m_axi_mm2s_rresp, m_axi_mm2s_rlast, m_axi_mm2s_rvalid, m_axi_mm2s_rready, mm2s_prmry_reset_out_n, m_axis_mm2s_tdata, m_axis_mm2s_tkeep, m_axis_mm2s_tvalid, m_axis_mm2s_tready, m_axis_mm2s_tlast, mm2s_cntrl_reset_out_n, m_axis_mm2s_cntrl_tdata, m_axis_mm2s_cntrl_tkeep, m_axis_mm2s_cntrl_tvalid, m_axis_mm2s_cntrl_tready, m_axis_mm2s_cntrl_tlast		If the parameter is assigned a value of zero, the output ports are tied to 0, and the input ports are left open.
C_M_AXI_MM2S_ADDR_ WIDTH	m_axi_mm2s_araddr		The setting of the parameter sets the vector width of the port.
C_M_AXI_MM2S_DATA_ WIDTH	m_axi_mm2s_rdata		The setting of the parameter sets the vector width of the port.
C_M_AXIS_MM2S_TDATA_ WIDTH	m_axis_mm2s_tdata, m_axis_mm2s_tkeep		The setting of the parameter sets the vector width of the port.

Table 4-17: **Parameter - I/O Signal Dependencies *(Cont'd)***

Parameter Name	Affects Signal	Depends on Parameter	Relationship Description
C_INCLUDE_S2MM	m_axi_s2mm_awaddr, m_axi_s2mm_awlen, m_axi_s2mm_awsize, m_axi_s2mm_awburst, m_axi_s2mm_awprot, m_axi_s2mm_awcache, m_axi_s2mm_awvalid, m_axi_s2mm_awready, m_axi_s2mm_wdata, m_axi_s2mm_wstrb, m_axi_s2mm_wlast, m_axi_s2mm_wvalid, m_axi_s2mm_wready, m_axi_s2mm_bresp, m_axi_s2mm_bvalid, m_axi_s2mm_bready, s2mm_prmry_reset_out_n, s_axis_s2mm_tdata, s_axis_s2mm_tkeep, s_axis_s2mm_tvalid, s_axis_s2mm_tready, s_axis_s2mm_tlast, s2mm_sts_reset_out_n, s_axis_s2mm_sts_tdata, s_axis_s2mm_sts_tkeep, s_axis_s2mm_sts_tvalid, s_axis_s2mm_sts_tready, s_axis_s2mm_sts_tlast		If the parameter is assigned a value of zero, the output ports are tied to 0 and the input ports are left open.
C_M_AXI_S2MM_ADDR_WIDTH	m_axis_s2mm_awaddr		The setting of the parameter sets the vector width of the port.
C_M_AXI_S2MM_DATA_WIDTH	m_axi_s2mm_wdata, m_axi_s2mm_wstrb		The setting of the parameter sets the vector width of the port.
C_S_AXIS_S2MM_TDATA_WIDTH	s_axis_s2mm_tdata, s_axis_s2mm_tkeep		The setting of the parameter sets the vector width of the port.

Parameter Descriptions

C_S_AXI_LITE_ADDR_WIDTH

- **Type:** Integer
- **Allowed Values:** 32 (default = 32)
- **Definition:** Address bus width of attached AXI on the AXI4-Lite interface
- **Description:** This integer parameter is used by the AXI4-Lite interface to size the AXI read and write address bus related components within the Lite interface. The EDK tool suite assigns this parameter a fixed value of 32.

C_S_AXI_LITE_DATA_WIDTH

- **Type:** Integer
- **Allowed Values:** 32 (default = 32)
- **Definition:** Data bus width of attached AXI on the AXI4-Lite interface
- **Description:** This integer parameter is used by the AXI4-Lite interface to size the AXI read and write data bus related components within the Lite interface. The EDK tool suite assigns this parameter a fixed value of 32.

C_DLYTMR_RESOLUTION

- **Type:** Integer
- **Allowed Values:** 1 to 100,000 (default = 125)
- **Definition:** Interrupt Delay Timer Resolution in AXI Secondary Clock cycles
- **Description:** This integer parameter is used to set the resolution of the Interrupt Delay Timer. Values specify the number of `m_axi_sg_aclk clock` cycles when C_INCLUDE_SG = 1 and `axi_lite_aclk` clock cycles when C_INCLUDE_SG = 0 between each tick of the delay timer.

C_PRMRY_IS_ACLK_ASYNC

- **Type:** Integer
- **Allowed Values:** 0,1 (default = 0)
- **Definition:** 0 = `s_axi_lite_aclk, m_axi_sg_aclk, m_axi_mm2s_aclk,` and `m_axi_s2mm_aclk` are synchronous to each other; 1 = `s_axi_lite_aclk, m_axi_sg_aclk, m_axi_mm2s_aclk,` and `m_axi_s2mm_aclk` are asynchronous to each other
- **Description:** Provides ability to operate the primary datapath asynchronously to the AXI4-Lite and Scatter/Gather Engine. This is used for applications where there is a requirement to operate the primary datapath at high frequencies, but this same high frequency requirement is not required for reading and writing control registers or for fetching and updating descriptors. In some cases, this allows for easier placement and timing closure at system build time. The EDK tool suite assigns this parameter automatically based on the `s_axi_lite_aclk`, `m_axi_sg_aclk`, `m_axi_mm2s_aclk`, and `m_axi_s2mm_aclk` clock sources.

C_S_AXI_LITE_ACLK_FREQ_HZ

- **Type:** Integer
- **Allowed Values:** All integer values
- **Definition:** Frequency in hertz of the s_axi_lite_aclk clock input.
- **Description:** This integer parameter is used by AXI DMA to correctly configure clock domain crossing logic. This parameter is only used when C_PRMRY_IS_ACLK_ASYNC = 1. The EDK tool suite assigns this parameter based on the clock frequency of the `s_axi_lite_aclk` source. When AXI DMA configured for asynchronous mode (C_PRMRY_IS_ACLK_ASYNC = 1) `s_axi_lite_aclk` frequency must be less than or equal to m_axi_sg_aclk frequency or undefined results occur.

C_M_AXI_SG_ACLK_FREQ_HZ

- **Type:** Integer
- **Allowed Values:** All integer values
- **Definition:** Frequency in hertz of the `m_axi_sg_aclk` clock input.
- **Description:** This integer parameter is used by AXI DMA to correctly configure clock domain crossing logic. This parameter is only used when C_PRMRY_IS_ACLK_ASYNC = 1 and C_INCLUDE_SG = 1. The EDK tool suite assigns this parameter based on the clock frequency of the `m_axi_sg_aclk` source.

C_M_AXI_MM2S_ACLK_FREQ_HZ

- **Type:** Integer
- **Allowed Values:** All integer values
- **Definition:** Frequency in hertz of the `axi_mm2s_aclk` clock input.
- **Description:** This integer parameter is used by AXI DMA to correctly configure clock domain crossing logic. This parameter is only used when C_PRMRY_IS_ACLK_ASYNC = 1. The EDK tool suite assigns this parameter based on the clock frequency of the `axi_mm2s_aclk` source.

C_M_AXI_S2MM_ACLK_FREQ_HZ

- **Type:** Integer
- **Allowed Values:** All integer values
- **Definition:** Frequency in hertz of the `axi_s2mm_aclk` clock input.
- **Description:** This integer parameter is used by AXI DMA to correctly configure clock domain crossing logic. This parameter is only used when C_PRMRY_IS_ACLK_ASYNC = 1. The EDK tool suite assigns this parameter based on the clock frequency of the `axi_s2mm_aclk` source.

C_M_AXI_SG_DATA_WIDTH

- **Type:** Integer
- **Allowed Values:** 32 (default = 32)
- **Definition:** Data bus width of attached AXI on the AXI Scatter/Gather interface
- **Description:** This integer parameter is used by the AXI Scatter/Gather interface to size the AXI read and write data bus related components within the Scatter/Gather Engine. The EDK tool suite assigns this parameter a fixed value of 32.

C_M_AXI_SG_ADDR_WIDTH

- **Type:** Integer
- **Allowed Values:** 32 (default = 32)
- **Definition:** Address bus width of attached AXI on the AXI Scatter Gather interface
- **Description:** This integer parameter is used by the AXI Scatter Gather interface to size the AXI read and write address bus related components within the Scatter Gather Engine. The EDK tool suite assigns this parameter a fixed value of 32.

C_INCLUDE_SG

- **Type:** Integer
- **Allowed Values:** 0,1 (default = 1)
- **Definition:** 0 = Exclude Scatter / Gather Channel; 1 = Include Scatter / Gather Channel
- **Description:** Include or exclude Scatter / Gather Channel. Setting this parameter to 0 configures the AXI DMA for Simple DMA Mode. Setting the parameter to 0 also causes all ports for the Scatter / Gather engine to be tied to zero and all of the input ports for the engine to be left open. Setting this parameter to 1 configures the AXI DMA for Scatter / Gather mode.

C_SG_INCLUDE_DESC_QUEUE

- **Type:** Integer
- **Allowed Values:** 0, 1(default = 0)
- **Definition:** 0 = Exclude Scatter Gather Descriptor Queuing; 1 = Include Scatter Gather Descriptor Queuing
- **Description:** Descriptor queuing allows multiple descriptors to be fetched and queued increasing the AXI DMA overall throughput. This feature uses First In First Out (FIFO)-based queues for fetching descriptors and updating descriptors providing a minimal bubble (typically 1 clock) continuous stream on primary datapaths.

  For lower performance applications, descriptor queueing can be excluded to save FPGA resources. If descriptor queueing is turned off, each descriptor is processed one at a time. In other words, a descriptor is fetched from remote memory, the transfer is performed, and then the descriptor is updated to remote memory. Then the next descriptor is fetched and the process continues.

  Note: Excluding Descriptor queues produces multi-clock dead cycles on the primary AXI4-Stream datapath between packets. Depending on descriptor-to-packet relationships can cause dead cycles within a packet.

Note: This parameter is used only when AXI DMA is configured for Scatter / Gather Mode, C_INCLUDE_SG = 1.

C_SG_INCLUDE_STSCNTRL_STRM

- **Type:** Integer
- **Allowed Values:** 0, 1(default = 1)
- **Definition:** 0 = Exclude Scatter Gather Status and Control Stream; 1 = Include Scatter Gather Status and Control Stream
- **Description:** The AXI Status and Control streams provide a method for transferring status packets and control packets between the AXI DMA engine and the stream client (that is, IP using AXI DMA and connected to MM2S and S2MM streams). In legacy systems, this information was transferred in the headers and footers of the primary datapaths, using up bandwidth. This option in AXI DMA allows the low-bandwidth metadata to be transferred separately from the primary data.

  The AXI Control Stream outputs control data associated with the Memory Map to Stream channel (MM2S), and the AXI Status Stream allows input of status data associated with the Stream to Memory Map channel (S2MM).

  Note: This parameter used only when AXI DMA is configured for Scatter / Gather Mode, C_INCLUDE_SG = 1.

C_S_AXIS_S2MM_STS_TDATA_WIDTH

- **Type:** Integer
- **Allowed Values:** 32 (default = 32)
- **Definition:** Data bus width of attached AXI on the AXI Status Stream interface
- **Description:** This integer parameter is used by the AXI Status Stream interface to size the data bus related components within AXI DMA. The EDK tool suite assigns this parameter a fixed value of 32.C_M_AXIS_MM2S_CNTRL_TDATA_WIDTH
- **Type:** Integer
- **Allowed Values:** 32 (default = 32)
- **Definition:** Data bus width of attached AXI on the AXI Control Stream interface
- **Description:** This integer parameter is used by the AXI Control Stream interface to size the data bus related components within AXI DMA. The EDK tool suite assigns this parameter a fixed value of 32.

C_SG_USE_STSAPP_LENGTH

- **Type:** Integer
- **Allowed Values:** 0, 1 (default = 1)
- **Definition:** 0 = Do not use receive length from APP4 of AXI Status Stream; 1 = Use receive length field in APP4 of AXI Status Stream
- **Description**: This parameter indicates whether to use the receive length field from the AXI Status Stream user application status packet. This parameter enables the use of a receive length from the status stream. The last word of the status packet (APP4) is captured by the AXI DMA engine for use in queuing up S2MM transfers prior to receipt of the entire packet. This prevents shortfalls and allows for exact byte transfers. Shortfalls occur when the actual amount of data being received on the S2MM channel is less than what was commanded to be transferred in the AXI DMA. In some use cases, like Ethernet, exact receive byte counts are not always known. In these shortfall cases, the AXI DataMover, the data movement engine of AXI DMA, is required to use store and forward logic to properly post transfer requests on the S2MM Memory Write interface. This causes multiple dead bus cycles to be inserted between packets, reducing overall throughput. If a receive length in bytes can be provided to the AXI DMA prior to receiving the packet, exact bytes to transfer can be commanded of the AXI DataMover allowing no wasted cycles. This feature is parameterizable to allow for applications that are unable to provide receive byte counts.

  Note: This parameter is used only when AXI DMA is configured for Scatter / Gather Mode, C_INCLUDE_SG = 1.

C_SG_LENGTH_WIDTH

- **Type:** Integer
- **Allowed Values:** 8 to 23 (default = 14)
- **Definition:** Width of length field in descriptor and in Status App4 field.
- **Description:** This parameter specifies the number of valid bits in the Buffer Length field and Transferred Bytes field of the descriptor.

  For S2MM Channel with C_SG_INCLUDE_STSCNTRL_STRM = 1 and C_SG_USE_STSAPP_LENGTH = 1, C_SG_LENGTH_WIDTH specifies the number of least significant bits of the last status word that are allocated for the receive length. The remainder of the bits in the last word can be used for other user specific application data if so desired. For AXI Ethernet applications, only 13 bits are used (the default number), but AXI DMA supports up to 23 bits for length, or 8 Mbytes. Reducing the value of this parameter reduces FPGA resource requirements.

  For Simple DMA Mode (C_INCLUDE_SG = 0) C_SG_LENGTH_WIDTH specifies the number of least significant bits of the MM2S_Length (offset 0x28) and S2MM_Length (offset 0x58) registers are valid.

C_INCLUDE_MM2S

- **Type:** Integer
- **Allowed Values:** 0,1 (default = 1)
- **Definition:** 0 = Exclude MM2S Channel; 1 = Include MM2S Channel
- **Description:** Include or exclude MM2S Channel. Setting this parameter to 0 causes all output ports for the MM2S channel to be tied to zero and all of the input ports for the respective channel to be left open.

  Note: Setting both C_INCLUDE_MM2S = 0 and C_INCLUDE_S2MM = 0 disables all logic within the AXI DMA and is not a valid configuration.

C_INCLUDE_S2MM

- **Type:** Integer
- **Allowed Values:** 0,1 (default = 1)
- **Definition:** 0 = Exclude S2MM Channel; 1 = Include S2MM Channel
- **Description:** Include or exclude S2MM Channel. Setting this parameter to 0 causes all output ports for the S2MM channel to be tied to zero, and all of the input ports for the respective channel to be left open.

  Note: Setting both C_INCLUDE_MM2S = 0 and C_INCLUDE_S2MM = 0 disables all logic within the AXI DMA and is not a valid configuration.

C_INCLUDE_MM2S_DRE

- **Type:** Integer
- **Allowed Values:** 0,1 (default = 0)
- **Definition:** 0 = Exclude MM2S Data Realignment Engine; 1 = Include MM2S Data Realignment Engine
- **Description:** Include or exclude MM2S Data Realignment Engine. For use cases where all transfers are C_M_AXIS_MM2S_TDATA_WIDTH aligned, this parameter can be set to 0 to exclude DRE-saving FPGA resources. Setting this parameter to 1 allows data realignment to the byte (8 bits) level on the primary memory map datapaths.

  For the MM2S channel, data is read from memory. If C_INCLUDE_MM2S_DRE = 1, 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. What is considered aligned or unaligned is based on the stream data width C_M_AXIS_MM2S_TDATA_WIDTH.

  For example, if C_M_AXIS_MM2S_TDATA_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 C_M_AXIS_MM2S_TDATA_WIDTH = 64, data is aligned if it is located at double-word offsets (64-bit offsets), that is 0x0, 0x8, 0x10, 0x18, and so forth.

  Note: MM2S Data Realignment Engine is excluded for data width greater than 64 Bits (C_M_AXIS_MM2S_TDATA_WIDTH > 64).

  Note: If DRE is disabled (C_INLUDE_MM2S_DRE = 0) for the respective channel, unaligned Buffer Addresses are not supported. Having an unaligned Buffer Address with DRE disabled produces undefined results. DRE Support is only available for AXI4-Stream a data width setting of 64-bits and under.

C_INCLUDE_S2MM_DRE

- **Type:** Integer
- **Allowed Values:** 0,1 (default = 0)
- **Definition:** 0 = Exclude S2MM Data Realignment Engine, 1 = Include S2MM Data Realignment Engine
- **Description:** Include or exclude S2MM Data Realignment Engine. For designs in which all transfers are C_S_AXIS_S2MM_TDATA_WIDTH aligned, this parameter can be set to 0 to exclude DRE-saving FPGA resources. Setting this parameter to 1 allows data realignment to the byte (8 bits) level on the primary memory map datapaths.

  For the S2MM channel, data is written to memory. If C_INCLUDE_S2MM_DRE = 1, data writes can start at any Buffer Address byte offset. The first byte in an AXI4-Stream is re-aligned to any buffer address byte offset.

  Note: S2MM Data Realignment Engine is excluded for a data width greater than 64 Bits (C_S_AXIS_S2MM_TDATA_WIDTH > 64).

  Note: If DRE is disabled (C_INCLUDE_S2MM_DRE = 0) for the respective channel, unaligned Buffer Addresses are not supported. Having an unaligned Buffer Address with DRE disabled produces undefined results. DRE Support is only available for an AXI4-Stream data width setting of 64-bits and under.

C_M_AXI_MM2S_ADDR_WIDTH

- **Type:** Integer
- **Allowed Values:** 32 (default = 32)
- **Definition:** Address bus width of attached AXI on the AXI MM2S Memory Map Read interface
- **Description:** This integer parameter is used by the MM2S interface to size the AXI read address bus-related components within the MM2S Channel. The EDK tool suite assigns this parameter a fixed value of 32.

C_M_AXI_MM2S_DATA_WIDTH

- **Type:** Integer
- **Allowed Values:** 32, 64, 128, 256 (default = 32)
- **Definition:** Data bus width of attached AXI on the AXI MM2S Memory Map Read interface
- **Description:** This integer parameter is used by the MM2S interface to size the AXI read data bus related components within the MM2S Channel. The EDK tools ensure correct sizing of the AXI data width based on EDK system configuration.

C_M_AXIS_MM2S_TDATA_WIDTH

- **Type:** Integer
- **Allowed Values:** 8, 16, 32, 64, 128, 256 (default = 32)
- **Definition:** Data bus width of attached AXI on the AXI MM2S Master Stream interface
- **Description:** This integer parameter is used by the MM2S interface to size the AXI Master Stream data bus-related components within the MM2S Channel.

  Note: This parameter must be set equal to C_M_AXI_MM2S_DATA_WIDTH.

C_M_AXI_S2MM_ADDR_WIDTH

- **Type:** Integer
- **Allowed Values:** 32 (default = 32)
- **Definition:** Address bus width of attached AXI on the AXI S2MM Memory Map Write interface
- **Description:** This integer parameter is used by the S2MM interface to size the AXI write address bus-related components within the S2MM Channel. The EDK tool suite assigns this parameter a fixed value of 32.

C_M_AXI_S2MM_DATA_WIDTH

- **Type:** Integer
- **Allowed Values:** 32, 64, 128, 256 (default = 32)
- **Definition:** Data bus width of attached AXI on the AXI S2MM Memory Map Write interface
- **Description:** This integer parameter is used by the S2MM interface to size the AXI write data bus-related components within the S2MM Channel. The EDK tools ensure correct sizing of the AXI data width based on EDK system configuration.

C_S_AXIS_S2MM_TDATA_WIDTH

- **Type:** Integer
- **Allowed Values:** 8, 16, 32, 64, 128, 256 (default = 32)
- **Definition:** Data bus width of attached AXI on the AXI S2MM Slave Stream interface
- **Description:** This integer parameter is used by the S2MM interface to size the AXI Slave Stream data bus-related components within the S2MM Channel.

  Note: This parameter must be set equal to C_M_AXI_S2MM_DATA_WIDTH.

C_MM2S_BURST_SIZE

- **Type:** Integer
- **Allowed Values:** 16, 32, 64, 128, 256 (default = 16)
- **Definition:** MM2S maximum burst size in data beats
- **Description:** Maximum burst size of the MM2S memory map interface. This parameter sets the granularity of burst partitioning. For example, if the burst size is set to 16, the maximum burst on the memory map interface is 16. Smaller values reduce throughput but result in less impact on the AXI infrastructure. Larger values increase throughput but result in a greater impact on the AXI infrastructure.

C_S2MM_BURST_SIZE

- **Type:** Integer
- **Allowed Values:** 16, 32, 64, 128, 256 (default = 16)
- **Definition:** S2MM maximum burst size in data beats
- **Description:** Maximum burst size of the S2MM memory map interface. This parameter sets the granularity of burst partitioning. For example, if the burst size is set to 16, the maximum burst on the memory map interface is 16. Smaller values reduce throughput but result in less impact on the AXI infrastructure. Larger values increase throughput but result in a greater impact on the AXI infrastructure.

Clock Domains

AXI DMA provides two clocking modes of operation: asynchronous and synchronous. Setting C_PRMRY_IS_ACLK_ASYNC = 1 enables this mode and creates four clock domains, as shown in Figure 4-25. 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. This parameter is set automatically by the EDK tool suite based on the clock sources feeding AXI DMA.

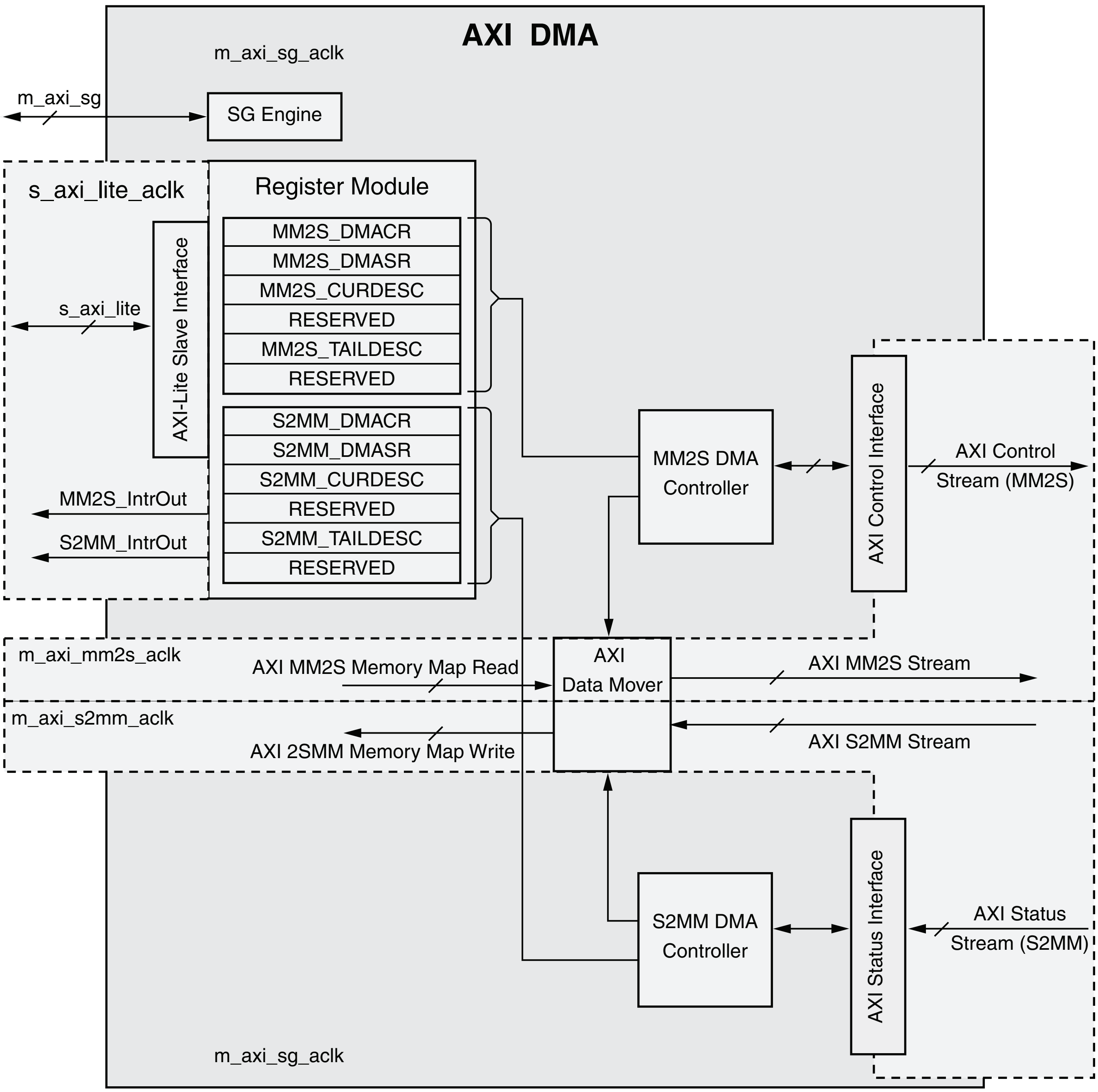


Figure 4-25: **Asynchronous Mode Clock Domains**

In synchronous mode (C_PRMRY_IS_ACLK_ASYNC = 0), all logic runs in a single clock domain. `s_axi_lite_aclk, m_axi_sg_aclk, m_axi_mm2s_aclk and m_axi_s2mm_aclk` must be tied to the same source. In asynchronous mode (C_PRMRY_IS_ACLK_ASYNC = 1) 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`.

In synchronous mode, C_PRMRY_IS_ACLK_ASYNC = 0, all logic runs in a single clock domain. `s_axi_lite_aclk`, `m_axi_sg_aclk, m_axi_mm2s_aclk`, and `m_axi_s2mm_aclk` must be tied to the same source otherwise undefined results occur.

Four parameters are provided to specify to AXI DMA the frequency of all of the clock sources, C_S_AXI_LITE_ACLK_FREQ_HZ, C_M_AXI_SG_ACLK_FREQ_HZ, C_M_AXI_MM2S_ACLK_FREQ_HZ, and C_M_AXI_S2MM_ACLK_HZ. AXI DMA uses these parameters to determine the necessary logic required to cross internal signals between the four clock domains. If using the EDK tool suite these parameters are set automatically. In synchronous mode the frequency hertz parameters are not used.

If AXI DMA is configured for Simple DMA Mode (C_INCLUDE_SG = 0) then the Scatter Gather clock input, `m_axi_sg_aclk`, is ignored and `s_axi_lite_aclk` is used to clock the DMA controllers, register block, and control interface.

The relationship of which signals and signal sets are clocked by what clock in asynchronous mode is shown in Table 4-18.

Table 4-18: **Asynchronous Mode Clock Distribution (C_PRMRY_IS_ACLK_ASYNC=1)**

Clock Source	I/O Ports (C_INCLUDE_SG = 1)	I/O Ports (C_INCLUDE_SG = 0)
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	All m_axi_sg_* Signals	N/A
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

XILINX

Chapter 5

Constraining the Core

There are no applicable constraints for this core.

Detailed Example Design

See Figure 1-2 and Figure 1-3 for an example design.

Additional Resources

Xilinx Resources

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

http://www.xilinx.com/support.

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

http://www.xilinx.com/support/documentation/sw_manuals/glossary.pdf.

List of Acronyms

Table 7-1: **List of Acronyms**

Acronym	Description
APP	Application
AXI	Advanced eXtensible Interface
CPU	Central Processing Unit
DMA	Direct Memory Access
DRE	Data Realignment Engine
DSP	Digital Signal Processing
EDK	Embedded Development Kit
EOF	End Of Frame
FF	Flip-Flop
FIFO	First In First Out
FPGA	Field Programmable Gate Array
HDL	Hardware Description Language
IOC	Interrupt On Complete
I/O	Input Output
IP	Intellectual Property
IRQ	Interrupt Request
ISE®	Integrated Software Environment
LUT	Lookup Table

Table 7-1: **List of Acronyms *(Cont'd)***

Acronym	Description
MHz	Mega Hertz
MM2S	Memory Map to Stream
MSB	Most Significant Bit
R/WC	Read / Write to Clear
RAM	Random Access Memory
RO	Read Only
RXEOF	Receive End Of Frame
RXSOF	Receive Start Of Frame
S2MM	Stream to Memory Map
SG	Scatter Gather
SOF	Start Of Frame
STS	Status Stream
TXEOF	Transmit End Of Frame
TXSOF	Transmit Start Of Frame
VDMA	Video Direct Memory Access
VHDL	VHSIC Hardware Description Language (VHSIC an acronym for Very High-Speed Integrated Circuits)
XPS	Xilinx Platform Studio (part of the EDK software)
XST	Xilinx Synthesis Technology

Solution Centers

See the Xilinx 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.

References

The *AMBA AXI4-Stream Protocol Specification* provides supplemental material useful with this product guide.

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

Technical Support

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

See the IP Release Notes Guide (XTP025) for more information on this core. For each core, there is a master Answer Record that contains the Release Notes and Known Issues list for the core being used. The following information is listed for each version of the core:

- New Features
- Resolved Issues
- Known Issues

Ordering Information

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

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

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
10/19/11	1.0	Initial Xilinx release.

Notice of Disclaimer