AXI Multichannel Direct Memory Access v1.0

LogiCORE IP Product Guide

Vivado Design Suite

PG288 October 4, 2017

XILINX
ALL PROGRAMMABLE™

Table of Contents

Introduction

The Xilinx® LogiCORE™ IP AXI Multichannel Direct Memory Access (AXI MCDMA) core is a soft Xilinx IP core for use with the Xilinx Vivado® Design Suite. The AXI MCDMA provides high-bandwidth direct memory access between memory and AXI4-Stream target peripherals. The AXI MCDMA core provides Scatter Gather (SG) interface with multiple channel support with independent configuration.

Features

- AXI4 Compliant
- AXI4 data width support of 32, 64, 128, 256, 512 and 1,024 bits
- AXI4-Stream data width support of 8, 16, 32, 64, 128, 256, 512 and 1,024 bits
- Supports up to 16 independent channels
- Supports per Channel Interrupt output
- Supports data realignment engine (DRE) alignment for streaming data width of up to 512 bits
- Supports up to 64 MB transfer per Buffer Descriptor (BD)
- Optional AXI4-Stream Control and Status Streams

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family(1)	UltraScale+™ UltraScale™ Zynq®-7000 All Programmable SoC, 7 Series FPGAs
Supported User Interfaces	AXI4-Lite, AXI4-Stream, AXI4
Resources	Performance and Resource Utilization web page
Provided with Core	
Design Files	VHDL
Example Design	VHDL
Test Bench	VHDL
Constraints File	Delivered at the time of IP generation
Simulation Model	Source HDL
Supported S/W Drivers(2)	Standalone and Linux
Tested Design Flows(3)	
Design Entry	Vivado Design Suite
Simulation	For supported simulators, see the Xilinx Design Tools: Release Notes Guide.
Synthesis	Vivado Synthesis
Support	
Provided by Xilinx at the Xilinx Support web page.	

Notes:

1. For a complete list of supported devices, see the Vivado IP catalog.
2. Standalone driver information can be found in the Software Development Kit (SDK) installation directory. See xilinx_drivers.htm in

   <install_directory>/SDK/<release>/data/embeddedsw/doc/xilinx_drivers.htm.

   Linux OS and driver support information is available from the Xilinx Wiki page.
3. For the supported versions of the tools, see the Xilinx Design Suite: Release Notes Guide.

Overview

Feature Summary

- AXI4 compliant
- Scatter/Gather support
  - Provides offloading of MCDMA management work from the CPU.
  - Provides fetch and update of Buffer Descriptors (BD) 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.
  - Allows each channel to have its own descriptor queue.
- Primary AXI4 data width support of 32, 64, 128, 256, 512 and 1,024 bits.
- DRE support for AXI4-Stream data with up to 512 bits.
- Up to 64 MB data transfer per descriptor.
- Optional AXI Control and Status Streams to interface to the AXI Ethernet IP.
  - Provides optional Control Stream for the MM2S Channel and Status Stream for the S2MM channel to offload low-bandwidth control and status from the high-bandwidth datapath.
- Provides up to 16 multiple channels of data movement each on MM2S and S2MM paths.
  - Supports independent channel configuration and status.
  - Supports per channel interrupt.
- Supports per channels packets statistics (packets drop, packets completion, etc).
- Features available on MM2S and S2MM side:
  - MM2S: Allows user-defined management of buffer descriptors.
  - MM2S: Select between Strict Priority, Weighted Round Robin (WRR) and Weighted Round Robin-Fair Distribution (WRR-FD) type of scheduler on MM2S side.
  - S2MM: Packets are dropped when MCDMA runs of Buffer Descriptors or drops the packets by disabling a channel.

Applications

The AXI MCDMA provides high-speed data movement between system memory and an AXI4-Stream-based target IP such as AXI Ethernet where packets have to be processed into separate descriptor queues based on TDEST value. AXI MCDMA can also be used in systems to replace multiple instances of AXI DMA.

Licensing and Ordering

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

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

Product Specification

The AXI Multichannel Direct Memory Access (AXI MCDMA) IP provides high-bandwidth direct memory access between the AXI4 memory mapped and AXI4-Stream IP interfaces. Its scatter gather capabilities also offload 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.

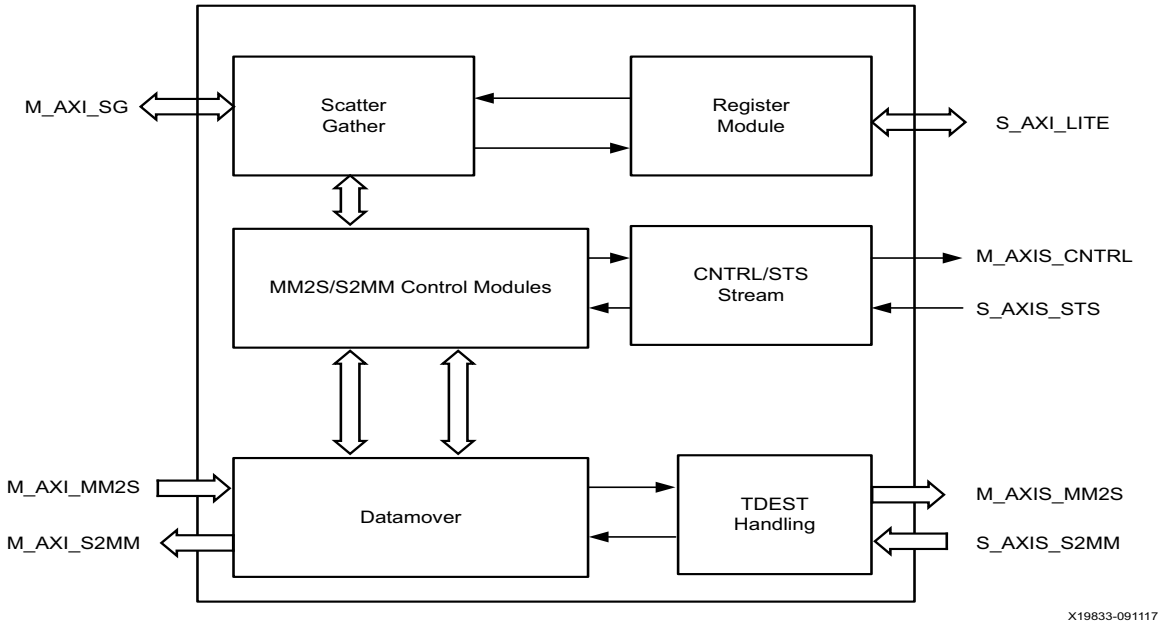

Figure 2-1: **Block Diagram**

Primary high-speed data movement between system memory and stream target is through the AXI4 Read Master to AXI4 memory-mapped to stream (MM2S) Master, and AXI stream to memory-mapped (S2MM) slave to AXI4 Write master. AXI MCDMA also enables up to 16 multiple channels of data movement on both MM2S and S2MM paths. The MM2S channel and S2MM channel operate independently.

Furthermore, the AXI MCDMA provides byte-level data realignment allowing memory reads and writes starting at any byte offset location. The MM2S channel supports an AXI Control stream for sending user application data to the target IP. For the S2MM channel, an AXI Status stream is provided for receiving user application data from the target IP. The channels are differentiated with the TDEST values present on the AXIS interface. A TDEST value of 0 corresponds to Channel 1, while a TDEST of 15 corresponds to Channel 16.

Performance

This section contains the following subsections.

- Maximum Frequencies
- Latency and Throughput

Maximum Frequencies

For full details about performance and resource utilization, visit the Performance and Resource Utilization web page.

Latency and Throughput

Table 2-1 provides the latency numbers for MM2S only configuration for synchronous mode. (`s_axi_lite_aclk` running at 60 MHz and `s_axi_aclk` running at 100 MHz).

Table 2-1: **MM2S Only Configuration For Synchronous Mode**

Description	Clocks cycles (s_axi_aclk)
Tail Descriptor register write to m_axi_sg_arvalid(1)	25
m_axi_sg_arvalid to m_axi_mm2s_arvalid(2)	27
m_axi_mm2s_arvalid to m_axis_mm2s_tvalid	6

Notes:
1. Time taken to fetch the first descriptor after the first TD is programmed.
2. Time taken to process the first descriptor

Table 2-2 provides the latency numbers for S2MM only configuration for synchronous mode. (`s_axi_lite_aclk` running at 60 MHz and `s_axi_aclk` running at 100 MHz).

Table 2-2: **S2MM Only Configuration For Synchronous Mode**

Description	Clocks cycles (s_axi_aclk)
Tail Descriptor register write to m_axi_sg_arvalid(1)	25
s_axis_s2mm_tvalid to m_axi_s2mm_awvalid(2)	50

Notes:
1. Time taken to fetch the first descriptor after the first Tail Descriptor (TD) is programmed.
2. Time taken to accept and process the first packet.

Resource Utilization

For full details about performance and resource utilization, visit the Performance and Resource Utilization web page.

Port Descriptions

Table 2-3: **AXI Port Descriptions**

Signal Name	Interface	Type	Init Value	Description
s_axi_aclk	Clock	Input		Clock input port, when the IP is configured in Sync Mode.
s_axi_lite_aclk	Clock	Input		AXI4-Lite clock input port.
m_axi_sg_aclk	Clock	Input		AXI MCDMA Scatter Gather clock. Available only when IP is configured for Async Mode.
m_axi_mm2s_aclk	Clock	Input		AXI MCDMA MM2S Primary Clock. Available when IP is configured in Async Mode.
m_axi_s2mm_aclk	Clock	Input		AXI MCDMA S2MM Primary clock. Available when IP is configured in Async Mode.
axi_resetn	Reset	Input		Active-Low AXI MCDMA reset. When asserted Low, it resets the AXI MCDMA. The reset has to be synchronous to s_axi_lite_aclk.
mm2s_ch*_introut	Interrupt	Output	0	Interrupt out for each channel of MM2S.
s2mm_ch*_introut	Interrupt	Output	0	Interrupt out for each channel of S2MM.
AXI4-Lite Interface Signals				
s_axi_lite_*	S_AXI_LITE	Input/ Output		See Appendix A of the *AXI Reference Guide* (UG1037) [Ref 9] for the AXI4 signal.
MM2S Memory Map Read Interface Signals				
m_axi_mm2s_*	M_AXI_MM2S	Input/ Output		See Appendix A of the AXI Reference Guide (UG1037) [Ref 9] for the AXI4 signal.
MM2S Master Stream Interface Signals				
mm2s_prmry_reset_out_n	M_AXIS_MM2S	Output	1	Primary MM2S Reset Out. Active-Low reset
m_axis_mm2s_*	M_AXIS_MM2S	Input/ Output		See Appendix A of the AXI Reference Guide (UG1037) [Ref 9] for the AXI4 signal.
MM2S Master Control Stream Interface Signals				
mm2s_cntrl_reset_out_n	M_AXIS_CNTRL	Output	1	Control Reset Out. Active-Low reset
m_axis_mm2s_cntrl_*	M_AXIS_CNTRL	Input/ Output		See Appendix A of the AXI Reference Guide (UG1037) [Ref 9] for the AXI4 signal.

Table 2-3: **AXI Port Descriptions *(Cont'd)***

Signal Name	Interface	Type	Init Value	Description
S2MM Memory Map Read Interface Signals				
m_axi_s2mm_*	M_AXI_S2MM	Input/ Output		See Appendix A of the AXI Reference Guide (UG1037) [Ref 9] for the AXI4 signal.
S2MM Slave Stream Interface Signals				
s2mm_prmry_reset_out_n	S_AXIS_S2MM	Output	1	Primary S2MM Reset Out. Active-Low reset
s_axis_s2mm_*	S_AXIS_SMM	Input/ Output		See Appendix A of the AXI Reference Guide (UG1037) [Ref 9] for the AXI4 signal.
S2MM Slave Status Stream Interface Signals				
s2mm_sts_reset_out_n	S_AXIS_STS	Output	1	Control Reset Out. Active-Low reset
s_axis_s2mm_sts_*	S_AXIS_STS	Input/ Output		See Appendix A of the AXI Reference Guide (UG1037) [Ref 9] for the AXI4 signal.
Scatter Gather Memory Map Interface Signals				
m_axi_sg_*	M_AXI_SG	Input/ Output		See Appendix A of the AXI Reference Guide (UG1037) [Ref 9] for the AXI4 signal.

Register Space

The MM2S AXI MCDMA core register space is shown in Table 2-4.

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

Table 2-4: **AXI MCDMA Core Register Space (MM2S)**

	Reg Offset	Name	Reg Description
Common MM2S Registers	0x000	MM2S_CCR	Common Control Register
	0x004	MM2S_CSR	Common Status Register
	0x008	MM2S_CHEN	Channel Enable/Disable
	0x00C	MM2S_CHSER	Channel In Progress Register
	0x010	MM2S_ERR	Error register
	0x014	MM2S_CH_SCHD_TYPE	Channel Queue Scheduler type
	0x018	MM2S_WRR_REG1	Weight of each channel (CH1-8)
	0x1C	MM2S_WRR_REG2	Weight of each channel (CH9-16)
	0x020	MM2S_CHANNELS_SERVICED	MM2S Channels Completed register
	0x024	MM2S_ARCACHE_ARUSER	Set the ARCACHE and ARUSER values for AXI4 read
	0x028	MM2S_INTR_STATUS	MM2S Channel Interrupt Monitor register
	0x02C-0x03C		RSVD
Channel 1	0x040	MM2S_CH1CR	CH1 Control Register
	0x044	MM2S_CH1SR	CH1 Status Register
	0x048	MM2S_CH1CURDESC_LSB	CH1 Current Descriptor (LSB)
	0x04C	MM2S_CH1CURDESC_MSB	CH1 Current Descriptor (MSB)
	0x050	MM2S_CH1TAILDESC_LSB	CH1 Tail Descriptor (LSB)
	0x054	MM2S_CH1TAILDESC_MSB	CH1 Tail Descriptor (MSB)
	0x058	MM2S_CH1PKTCOUNT_STAT	CH1 Packet Processed count
	0x05C-0x07C		RSVD

Table 2-4: **AXI MCDMA Core Register Space (MM2S) *(Cont'd)***

	Reg Offset	Name	Reg Description
Channel 2	0x080	MM2S_CH2CR	CH2 Control Register
	0x084	MM2S_CH2SR	CH2 Status Register
	0x088	MM2S_CH2CURDESC_LSB	CH2 Current Descriptor (LSB)
	0x08C	MM2S_CH2CURDESC_MSB	CH2 Current Descriptor (MSB)
	0x090	MM2S_CH2TAILDESC_LSB	CH2 Tail Descriptor (LSB)
	0x094	MM2S_CH2TAILDESC_MSB	CH2 Tail Descriptor (MSB)
	0x098	MM2S_CH2PKTCOUNT_STAT	CH2 Packet Processed count
	0x09C-0x0BC		RSVD
Channel 3	0x0C0	MM2S_CH3CR	CH3 Control Register
	0x0C4	MM2S_CH3SR	CH3 Status Register
	0x0C8	MM2S_CH3CURDESC_LSB	CH3 Current Descriptor (LSB)
	0x0CC	MM2S_CH3CURDESC_MSB	CH3 Current Descriptor (MSB)
	0x0D0	MM2S_CH3TAILDESC_LSB	CH3 Tail Descriptor (LSB)
	0x0D4	MM2S_CH3TAILDESC_MSB	CH3 Tail Descriptor (MSB)
	0x0D8	MM2S_CH3PKTCOUNT_STAT	CH3 Packet Processed count
	0x0DC-0x0FC		RSVD
Channel 4	0x100	MM2S_CH4CR	CH4 Control Register
	0x104	MM2S_CH4SR	CH4 Status Register
	0x108	MM2S_CH4CURDESC_LSB	CH4 Current Descriptor (LSB)
	0x10C	MM2S_CH4CURDESC_MSB	CH4 Current Descriptor (MSB)
	0x110	MM2S_CH4TAILDESC_LSB	CH4 Tail Descriptor (LSB)
	0x114	MM2S_CH4TAILDESC_MSB	CH4 Tail Descriptor (MSB)
	0x118	MM2S_CH4PKTCOUNT_STAT	CH4 Packet Processed count
	0x11C - 0x13C		RSVD
Channel 5	0x140	MM2S_CH5CR	CH5 Control Register
	0x144	MM2S_CH5SR	CH5 Status Register
	0x148	MM2S_CH5CURDESC_LSB	CH5 Current Descriptor (LSB)
	0x14C	MM2S_CH5CURDESC_MSB	CH5 Current Descriptor (MSB)
	0x150	MM2S_CH5TAILDESC_LSB	CH5 Tail Descriptor (LSB)
	0x154	MM2S_CH5TAILDESC_MSB	CH5 Tail Descriptor (MSB)
	0x158	MM2S_CH5PKTCOUNT_STAT	CH5 Packet Processed count
	0x15C-0x17C		RSVD

Table 2-4: **AXI MCDMA Core Register Space (MM2S) *(Cont'd)***

	Reg Offset	Name	Reg Description
Channel 6	0x180	MM2S_CH6CR	CH6 Control Register
	0x184	MM2S_CH6SR	CH6 Status Register
	0x188	MM2S_CH6CURDESC_LSB	CH6 Current Descriptor (LSB)
	0x18C	MM2S_CH6CURDESC_MSB	CH6 Current Descriptor (MSB)
	0x190	MM2S_CH6TAILDESC_LSB	CH6 Tail Descriptor (LSB)
	0x194	MM2S_CH6TAILDESC_MSB	CH6 Tail Descriptor (MSB)
	0x198	MM2S_CH6PKTCOUNT_STAT	CH6 Packet Processed count
	0x19C-0x1BC		RSVD
Channel 7	0x1C0	MM2S_CH7CR	CH7 Control Register
	0x1C4	MM2S_CH7SR	CH7 Status Register
	0x1C8	MM2S_CH7CURDESC_LSB	CH7 Current Descriptor (LSB)
	0x1CC	MM2S_CH7CURDESC_MSB	CH7 Current Descriptor (MSB)
	0x1D0	MM2S_CH7TAILDESC_LSB	CH7 Tail Descriptor (LSB)
	0x1D4	MM2S_CH7TAILDESC_MSB	CH7 Tail Descriptor (MSB)
	0x1D8	MM2S_CH7PKTCOUNT_STAT	CH7 Packet Processed count
	0x1DC-0x1FC		RSVD
Channel 8	0x200	MM2S_CH8CR	CH8 Control Register
	0x204	MM2S_CH8SR	CH8 Status Register
	0x208	MM2S_CH8CURDESC_LSB	CH8 Current Descriptor (LSB)
	0x20C	MM2S_CH8CURDESC_MSB	CH8 Current Descriptor (MSB)
	0x210	MM2S_CH8TAILDESC_LSB	CH8 Tail Descriptor (LSB)
	0x214	MM2S_CH8TAILDESC_MSB	CH8 Tail Descriptor (MSB)
	0x218	MM2S_CH8PKTCOUNT_STAT	CH8 Packet Processed count
	0x21C-0x23C		RSVD
Channel 9	0x240	MM2S_CH9CR	CH9 Control Register
	0x244	MM2S_CH9SR	CH9 Status Register
	0x248	MM2S_CH9CURDESC_LSB	CH9 Current Descriptor (LSB)
	0x24C	MM2S_CH9CURDESC_MSB	CH9 Current Descriptor (MSB)
	0x250	MM2S_CH9TAILDESC_LSB	CH9 Tail Descriptor (LSB)
	0x254	MM2S_CH9TAILDESC_MSB	CH9 Tail Descriptor (MSB)
	0x258	MM2S_CH9PKTCOUNT_STAT	CH9 Packet Processed count
	0x25C-0x27C		RSVD

Table 2-4: **AXI MCDMA Core Register Space (MM2S) *(Cont'd)***

	Reg Offset	Name	Reg Description
Channel 10	0x280	MM2S_CH10CR	CH10 Control Register
	0x284	MM2S_CH10SR	CH10 Status Register
	0x288	MM2S_CH10CURDESC_LSB	CH10 Current Descriptor (LSB)
	0x28C	MM2S_CH10CURDESC_MSB	CH10 Current Descriptor (MSB)
	0x290	MM2S_CH10TAILDESC_LSB	CH10 Tail Descriptor (LSB)
	0x294	MM2S_CH10TAILDESC_MSB	CH10 Tail Descriptor (MSB)
	0x298	MM2S_CH10PKTCOUNT_STAT	CH10 Packet Processed count
	0x29C-0x2BC		RSVD
Channel 11	0x2C0	MM2S_CH11CR	CH11 Control Register
	0x2C4	MM2S_CH11SR	CH11 Status Register
	0x2C8	MM2S_CH11CURDESC_LSB	CH11 Current Descriptor (LSB)
	0x2CC	MM2S_CH11CURDESC_MSB	CH11 Current Descriptor (MSB)
	0x2D0	MM2S_CH11TAILDESC_LSB	CH11 Tail Descriptor (LSB)
	0x2D4	MM2S_CH11TAILDESC_MSB	CH11 Tail Descriptor (MSB)
	0x2D8	MM2S_CH11PKTCOUNT_STAT	CH11 Packet Processed count
	0x2DC-0x2FC		RSVD
Channel 12	0x300	MM2S_CH12CR	CH12 Control Register
	0x304	MM2S_CH12SR	CH12 Status Register
	0x308	MM2S_CH12CURDESC_LSB	CH12 Current Descriptor (LSB)
	0x30C	MM2S_CH12CURDESC_MSB	CH12 Current Descriptor (MSB)
	0x310	MM2S_CH12TAILDESC_LSB	CH12 Tail Descriptor (LSB)
	0x314	MM2S_CH12TAILDESC_MSB	CH12 Tail Descriptor (MSB)
	0x318	MM2S_CH12PKTCOUNT_STAT	CH12 Packet Processed count
	0x31C-0x33C		RSVD
Channel 13	0x340	MM2S_CH13CR	CH13 Control Register
	0x344	MM2S_CH13SR	CH13 Status Register
	0x348	MM2S_CH13CURDESC_LSB	CH13 Current Descriptor (LSB)
	0x34C	MM2S_CH13CURDESC_MSB	CH13 Current Descriptor (MSB)
	0x350	MM2S_CH13TAILDESC_LSB	CH13 Tail Descriptor (LSB)
	0x354	MM2S_CH13TAILDESC_MSB	CH13 Tail Descriptor (MSB)
	0x358	MM2S_CH13PKTCOUNT_STAT	CH13 Packet Processed count
	0x35C-0x37C		RSVD

Table 2-4: **AXI MCDMA Core Register Space (MM2S) *(Cont'd)***

	Reg Offset	Name	Reg Description
Channel 14	0x380	MM2S_CH14CR	CH14 Control Register
	0x384	MM2S_CH14SR	CH14 Status Register
	0x388	MM2S_CH14CURDESC_LSB	CH14 Current Descriptor (LSB)
	0x38C	MM2S_CH14CURDESC_MSB	CH14 Current Descriptor (MSB)
	0x390	MM2S_CH14TAILDESC_LSB	CH14 Tail Descriptor (LSB)
	0x394	MM2S_CH14TAILDESC_MSB	CH14 Tail Descriptor (MSB)
	0x398	MM2S_CH14PKTCOUNT_STAT	CH14 Packet Processed count
	0x39C-0x3BC		RSVD
Channel 15	0x3C0	MM2S_CH15CR	CH15 Control Register
	0x3C4	MM2S_CH15SR	CH15 Status Register
	0x3C8	MM2S_CH15CURDESC_LSB	CH15 Current Descriptor (LSB)
	0x3CC	MM2S_CH15CURDESC_MSB	CH15 Current Descriptor (MSB)
	0x3D0	MM2S_CH15TAILDESC_LSB	CH15 Tail Descriptor (LSB)
	0x3D4	MM2S_CH15TAILDESC_MSB	CH15 Tail Descriptor (MSB)
	0x3D8	MM2S_CH15PKTCOUNT_STAT	CH15 Packet Processed count
	0x3DC-0x3FC		RSVD
Channel 16	0x400	MM2S_CH16CR	CH16 Control Register
	0x404	MM2S_CH16SR	CH16 Status Register
	0x408	MM2S_CH16CURDESC_LSB	CH16 Current Descriptor (LSB)
	0x40C	MM2S_CH16CURDESC_MSB	CH16 Current Descriptor (MSB)
	0x410	MM2S_CH16TAILDESC_LSB	CH16 Tail Descriptor (LSB)
	0x414	MM2S_CH16TAILDESC_MSB	CH16 Tail Descriptor (MSB)
	0x418	MM2S_CH16PKTCOUNT_STAT	CH16 Packet Processed count
	0x41C-0x43C		RSVD
Channel Groups	0x440	MM2S Channel Observer 1	Channel service information for channels selected in Vivado® Integrated Design Environment (IDE) under Group 1
	0x444	MM2S Channel Observer 2	Channel service information for channels selected in Vivado IDE under Group 2
	0x448	MM2S Channel Observer 3	Channel service information for channels selected in Vivado IDE under Group 3
	0x44C	MM2S Channel Observer 4	Channel service information for channels selected in Vivado IDE under Group 4
	0x450	MM2S Channel Observer 5	Channel service information for channels selected in Vivado IDE under Group 5
	0x454	MM2S Channel Observer 6	Channel service information for channels selected in Vivado IDE under Group 6

The S2MM AXI MCDMA core register space is shown in Table 2-5.

Table 2-5: **AXI MCDMA Core Register Space (S2MM)**

	Register Offset	Name	Register Description
Common S2MM Registers	0x500	S2MM_CCR	Common Control Register
	0x504	S2MM_CSR	Common Status Register
	0x508	S2MM_CHEN	Channel Enable/Disable
	0x50C	S2MM_CHSER	Channel In Progress Register
	0x510	S2MM_ERR	MCDMA Error Register
	0x514	S2MM_PKTDROP	S2MM Packet Drop Stat
	0x518	S2MM_ CHANNELS_SERVICED	S2MM Channels Completed register
	0x51C	S2MM_AWCACHE_AWUSER	Set the values for awcache and awuser ports
	0x520	S2MM_INTR_STATUS	S2MM Channel Interrupt Monitor register
	0x524-0x53C		RSVD
Channel 1	0x540	S2MM_CH1CR	CH1 Control Register
	0x544	S2MM_CH1SR	CH1 Status Register
	0x548	S2MM_CH1CURDESC_LSB	CH1 Current Descriptor (LSB)
	0x54C	S2MM_CH1CURDESC_MSB	CH1 Current Descriptor (MSB)
	0x550	S2MM_CH1TAILDESC_LSB	CH1 Tail Descriptor (LSB)
	0x554	S2MM_CH1TAILDESC_MSB	CH1 Tail Descriptor (MSB)
	0x558	S2MM_CH1PKTDROP_STAT	CH1 Packet Drop Stat
	0x55C	S2MM_CH1PKTCOUNT_STAT	CH1 Packet Processed count
	0x560-0x57C		RSVD
Channel 2	0x580	S2MM_CH2CR	CH2 Control Register
	0x584	S2MM_CH2SR	CH2 Status Register
	0x588	S2MM_CH2CURDESC_LSB	CH2 Current Descriptor (LSB)
	0x58C	S2MM_CH2CURDESC_MSB	CH2 Current Descriptor (MSB)
	0x590	S2MM_CH2TAILDESC_LSB	CH2 Tail Descriptor (LSB)
	0x594	S2MM_CH2TAILDESC_MSB	CH2 Tail Descriptor (MSB)
	0x598	S2MM_CH2PKTDROP_STAT	CH2 Packet Drop Stat
	0x59C	S2MM_CH2PKTCOUNT_STAT	CH2 Packet Processed count
	0x5A0-0x5BC		RSVD

Table 2-5: **AXI MCDMA Core Register Space (S2MM) *(Cont'd)***

	Register Offset	Name	Register Description
Channel 3	0x5C0	S2MM_CH3CR	CH3 Control Register
	0x5C4	S2MM_CH3SR	CH3 Status Register
	0x5C8	S2MM_CH3CURDESC_LSB	CH3 Current Descriptor (LSB)
	0x5CC	S2MM_CH3CURDESC_MSB	CH3 Current Descriptor (MSB)
	0x5D0	S2MM_CH3TAILDESC_LSB	CH3 Tail Descriptor (LSB)
	0x5D4	S2MM_CH3TAILDESC_MSB	CH3 Tail Descriptor (MSB)
	0x5D8	S2MM_CH3PKTDROP_STAT	CH3 Packet Drop Stat
	0x5DC	S2MM_CH3PKTCOUNT_STAT	CH3 Packet Processed count
	0x5E0-0x5FC		RSVD
Channel 4	0x600	S2MM_CH4CR	CH4 Control Register
	0x604	S2MM_CH4SR	CH4 Status Register
	0x608	S2MM_CH4CURDESC_LSB	CH4 Current Descriptor (LSB)
	0x60C	S2MM_CH4CURDESC_MSB	CH4 Current Descriptor (MSB)
	0x610	S2MM_CH4TAILDESC_LSB	CH4 Tail Descriptor (LSB)
	0x614	S2MM_CH4TAILDESC_MSB	CH4 Tail Descriptor (MSB)
	0x618	S2MM_CH4PKTDROP_STAT	CH4 Packet Drop Stat
	0x61C	S2MM_CH4PKTCOUNT_STAT	CH4 Packet Processed count
	0x620-0x63C		RSVD
Channel 5	0x640	S2MM_CH5CR	CH5 Control Register
	0x644	S2MM_CH5SR	CH5 Status Register
	0x648	S2MM_CH5CURDESC_LSB	CH5 Current Descriptor (LSB)
	0x64C	S2MM_CH5CURDESC_MSB	CH5 Current Descriptor (MSB)
	0x650	S2MM_CH5TAILDESC_LSB	CH5 Tail Descriptor (LSB)
	0x654	S2MM_CH5TAILDESC_MSB	CH5 Tail Descriptor (MSB)
	0x658	S2MM_CH5PKTDROP_STAT	CH5 Packet Drop Stat
	0x65C	S2MM_CH5PKTCOUNT_STAT	CH5 Packet Processed count
	0x660-0x67C		RSVD

Table 2-5: **AXI MCDMA Core Register Space (S2MM) *(Cont'd)***

	Register Offset	Name	Register Description
Channel 6	0x680	S2MM_CH6CR	CH6 Control Register
	0x684	S2MM_CH6SR	CH6 Status Register
	0x688	S2MM_CH6CURDESC_LSB	CH6 Current Descriptor (LSB)
	0x68C	S2MM_CH6CURDESC_MSB	CH6 Current Descriptor (MSB)
	0x690	S2MM_CH6TAILDESC_LSB	CH6 Tail Descriptor (LSB)
	0x694	S2MM_CH6TAILDESC_MSB	CH6 Tail Descriptor (MSB)
	0x698	S2MM_CH6PKTDROP_STAT	CH6 Packet Drop Stat
	0x69C	S2MM_CH6PKTCOUNT_STAT	CH6 Packet Processed count
	0x6A0-0x6BC		RSVD
Channel 7	0x6C0	S2MM_CH7CR	CH7 Control Register
	0x6C4	S2MM_CH7SR	CH7 Status Register
	0x6C8	S2MM_CH7CURDESC_LSB	CH7 Current Descriptor (LSB)
	0x6CC	S2MM_CH7CURDESC_MSB	CH7 Current Descriptor (MSB)
	0x6D0	S2MM_CH7TAILDESC_LSB	CH7 Tail Descriptor (LSB)
	0x6D4	S2MM_CH7TAILDESC_MSB	CH7 Tail Descriptor (MSB)
	0x6D8	S2MM_CH7PKTDROP_STAT	CH7 Packet Drop Stat
	0x6DC	S2MM_CH7PKTCOUNT_STAT	CH7 Packet Processed count
	0x6E0-0x6FC		RSVD
Channel 8	0x700	S2MM_CH8CR	CH8 Control Register
	0x704	S2MM_CH8SR	CH8 Status Register
	0x708	S2MM_CH8CURDESC_LSB	CH8 Current Descriptor (LSB)
	0x70C	S2MM_CH8CURDESC_MSB	CH8 Current Descriptor (MSB)
	0x710	S2MM_CH8TAILDESC_LSB	CH8 Tail Descriptor (LSB)
	0x714	S2MM_CH8TAILDESC_MSB	CH8 Tail Descriptor (MSB)
	0x718	S2MM_CH8PKTDROP_STAT	CH8 Packet Drop Stat
	0x71C	S2MM_CH8PKTCOUNT_STAT	CH8 Packet Processed count
	0x720-0x73C		RSVD

Table 2-5: **AXI MCDMA Core Register Space (S2MM) *(Cont'd)***

	Register Offset	Name	Register Description
Channel 9	0x740	S2MM_CH9CR	CH9 Control Register
	0x744	S2MM_CH9SR	CH9 Status Register
	0x748	S2MM_CH9CURDESC_LSB	CH9 Current Descriptor (LSB)
	0x74C	S2MM_CH9CURDESC_MSB	CH9 Current Descriptor (MSB)
	0x750	S2MM_CH9TAILDESC_LSB	CH9 Tail Descriptor (LSB)
	0x754	S2MM_CH9TAILDESC_MSB	CH9 Tail Descriptor (MSB)
	0x758	S2MM_CH9PKTDROP_STAT	CH9 Packet Drop Stat
	0x75C	S2MM_CH9PKTCOUNT_STAT	CH9 Packet Processed count
	0x760-0x77C		RSDV
Channel 10	0x780	S2MM_CH10CR	CH10 Control Register
	0x784	S2MM_CH10SR	CH10 Status Register
	0x788	S2MM_CH10CURDESC_LSB	CH10 Current Descriptor (LSB)
	0x78C	S2MM_CH10CURDESC_MSB	CH10 Current Descriptor (MSB)
	0x790	S2MM_CH10TAILDESC_LSB	CH10 Tail Descriptor (LSB)
	0x794	S2MM_CH10TAILDESC_MSB	CH10 Tail Descriptor (MSB)
	0x798	S2MM_CH10PKTDROP_STAT	CH10 Packet Drop Stat
	0x79C	S2MM_CH10PKTCOUNT_STAT	CH10 Packet Processed count
	0x7A0-0x7BC		RSVD
Channel 11	0x7C0	S2MM_CH11CR	CH11 Control Register
	0x7C4	S2MM_CH11SR	CH11 Status Register
	0x7C8	S2MM_CH11CURDESC_LSB	CH11 Current Descriptor (LSB)
	0x7CC	S2MM_CH11CURDESC_MSB	CH11 Current Descriptor (MSB)
	0x7D0	S2MM_CH11TAILDESC_LSB	CH11 Tail Descriptor (LSB)
	0x7D4	S2MM_CH11TAILDESC_MSB	CH11 Tail Descriptor (MSB)
	0x7D8	S2MM_CH11PKTDROP_STAT	CH11 Packet Drop Stat
	0x7DC	S2MM_CH11PKTCOUNT_STAT	CH11 Packet Processed count
	0x7E0-0x7FC		RSVD

Table 2-5: **AXI MCDMA Core Register Space (S2MM) *(Cont'd)***

	Register Offset	Name	Register Description
Channel 12	0x800	S2MM_CH12CR	CH12 Control Register
	0x804	S2MM_CH12SR	CH12 Status Register
	0x808	S2MM_CH12CURDESC_LSB	CH12 Current Descriptor (LSB)
	0x80C	S2MM_CH12CURDESC_MSB	CH12 Current Descriptor (MSB)
	0x810	S2MM_CH12TAILDESC_LSB	CH12 Tail Descriptor (LSB)
	0x814	S2MM_CH12TAILDESC_MSB	CH12 Tail Descriptor (MSB)
	0x818	S2MM_CH12PKTDROP_STAT	CH12 Packet Drop Stat
	0x81C	S2MM_CH12PKTCOUNT_STAT	CH12 Packet Processed count
	0x820-0x83C		RSVD
Channel 13	0x840	S2MM_CH13CR	CH13 Control Register
	0x844	S2MM_CH13SR	CH13 Status Register
	0x848	S2MM_CH13CURDESC_LSB	CH13 Current Descriptor (LSB)
	0x84C	S2MM_CH13CURDESC_MSB	CH13 Current Descriptor (MSB)
	0x850	S2MM_CH13TAILDESC_LSB	CH13 Tail Descriptor (LSB)
	0x854	S2MM_CH13TAILDESC_MSB	CH13 Tail Descriptor (MSB)
	0x858	S2MM_CH13PKTDROP_STAT	CH13 Packet Drop Stat
	0x85C	S2MM_CH13PKTCOUNT_STAT	CH13 Packet Processed count
	0x860-0x87C		RSVD
Channel 14	0x880	S2MM_CH14CR	CH14 Control Register
	0x884	S2MM_CH14SR	CH14 Status Register
	0x888	S2MM_CH14CURDESC_LSB	CH14 Current Descriptor (LSB)
	0x88C	S2MM_CH14CURDESC_MSB	CH14 Current Descriptor (MSB)
	0x890	S2MM_CH14TAILDESC_LSB	CH14 Tail Descriptor (LSB)
	0x894	S2MM_CH14TAILDESC_MSB	CH14 Tail Descriptor (MSB)
	0x898	S2MM_CH14PKTDROP_STAT	CH14 Packet Drop Stat
	0x89C	S2MM_CH14PKTCOUNT_STAT	CH14 Packet Processed count
	0x8A0-0x8BC		RSVD

Table 2-5: **AXI MCDMA Core Register Space (S2MM) *(Cont'd)***

	Register Offset	Name	Register Description
Channel 15	0x8C0	S2MM_CH15CR	CH15 Control Register
	0x8C4	S2MM_CH15SR	CH15 Status Register
	0x8C8	S2MM_CH15CURDESC_LSB	CH15 Current Descriptor (LSB)
	0x8CC	S2MM_CH15CURDESC_MSB	CH15 Current Descriptor (MSB)
	0x8D0	S2MM_CH15TAILDESC_LSB	CH15 Tail Descriptor (LSB)
	0x8D4	S2MM_CH15TAILDESC_MSB	CH15 Tail Descriptor (MSB)
	0x8D8	S2MM_CH15PKTDROP_STAT	CH15 Packet Drop Stat
	0x8DC	S2MM_CH15PKTCOUNT_STAT	CH15 Packet Processed count
	0x8E0-0x8FC		RSVD
Channel 16	0x900	S2MM_CH16CR	CH16 Control Register
	0x904	S2MM_CH16SR	CH16 Status Register
	0x908	S2MM_CH16CURDESC_LSB	CH16 Current Descriptor (LSB)
	0x90C	S2MM_CH16CURDESC_MSB	CH16 Current Descriptor (MSB)
	0x910	S2MM_CH16TAILDESC_LSB	CH16 Tail Descriptor (LSB)
	0x914	S2MM_CH16TAILDESC_MSB	CH16 Tail Descriptor (MSB)
	0x918	S2MM_CH16PKTDROP_STAT	CH16 Packet Drop Stat
	0x91C	S2MM_CH16PKTCOUNT_STAT	CH16 Packet Processed count
	0x920-0x93C		RSVD
Channel Groups	0x940	S2MM Channel Observer 1	Channel service information for channels selected in Vivado IDE under Group 1
	0x944	S2MM Channel Observer 2	Channel service information for channels selected in Vivado IDE under Group 2
	0x948	S2MM Channel Observer 3	Channel service information for channels selected in Vivado IDE under Group 3
	0x94C	S2MM Channel Observer 4	Channel service information for channels selected in Vivado IDE under Group 4
	0x950	S2MM Channel Observer 5	Channel service information for channels selected in Vivado IDE under Group 5
	0x954	S2MM Channel Observer 6	Channel service information for channels selected in Vivado IDE under Group 6

The common AXI MCDMA core register space is shown in Table 2-6.

Table 2-6: **Common AXI MCDMA Core Register Space**

	Register Offset	Name	Register Description
SG User/ Cache	0x4B0	SG_CACHE_USER	Set the values for cache and user ports (read and write)

MM2S Common Control Register (0x000)

The MM2S path has a Control register to control the MCDMA engine. This register allows you to Reset or put the MCDMA in Run mode.

Table 2-7: **MM2S Common Control Register (0x000)**

Bits	Name	Default Value	Access	Description
31:3	RSVD			
2	Multichannel MCDMA Reset (MCDMA.RST)	0	R/W	Soft reset for resetting the AXI MCDMA core. Setting this bit to a 1 causes the AXI MCDMA to be reset. Reset is accomplished gracefully. Pending commands/transfers are flushed or completed. 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. This resets all the channels and clears all the registers. Re-configure the MCDMA after a soft reset. Resetting the MM2S also resets the S2MM.

Table 2-7: **MM2S Common Control Register (0x000) *(Cont'd)***

Bits	Name	Default Value	Access	Description
1	RSVD			
0	Multichannel MCDMA RunStop (MCDMA.RS)	0	R/W	Start/Stop the Multichannel MM2s MCDMA channel. • 0 = Stop – MCDMA stops when current (if any) MCDMA operations are complete. For Scatter/Gather Mode pending commands/transfers are flushed or completed. Descriptors in the update queue are allowed to finish updating to remote memory before engine halt. Data integrity on MM2S AXI4 cannot be guaranteed. The halted bit in the MCDMA Status register asserts to 1 when the MCDMA engine is halted. This bit is cleared by AXI MCDMA hardware when an error occurs. The CPU can also choose to clear this bit to stop MCDMA operations. • 1 = Run – Start MCDMA operations. The halted bit in the MCDMA Status register deasserts to 0 when the MCDMA engine begins operations.

MM2S Common Status Register (0x004)

The MM2S path has a Common Status register to keep track of the MCDMA status. This register provides the overall status of the MCDMA engine.

Table 2-8: **MM2S Common Status Register (0x004)**

Bits	Name	Default Value	Access	Description
31:2	RSVD			
1	Idle	0	RO	MCDMA MM2S Idle. Indicates the state of AXI MCDMA operations. When IDLE, indicates the SG Engine has reached the tail pointer for all the channels and all queued descriptors have been processed. Writing to the tail pointer register to any channel automatically restarts MCDMA operations. • 0 = Not Idle. • 1 = Idle.
0	Halted (MCDMA.Halted)	1	RO	MCDMA Halted. Indicates the run/stop state of the MCDMA. • 0 = MCDMA channel running. • 1 = MCDMA.RS bit is set to 0. There can be a lag of time between when MCDMACR.RS = 0 and when MCDMASR.Halted = 1.

MM2S Channel Enable/Disable Register (0x008)

This register provides the capability to enable/disable a particular channel from being serviced.

Table 2-9: **MM2S Channel Enable/Disable Register (0x008)**

Bits	Name	Default Value	Access	Description
31:16	Reserved	0	R	NA
15:0	Channel Enable	0x1	R/W	Setting a bit to 1 enables the channel for service. Each bit corresponds to a channel. Bit [0] corresponds to Channel0, bit [1] to Channel1 and so on. Only those bits corresponding to the number of channels are programmable.

This register does not stop the BD fetch of the channel; it only stops a particular channel from being serviced. This register can be programmed at any time, but it will come into effect only when an end-of-frame (EOF) bit is detected on the BD while the scheduler is running.

Note: Disabling a channel does not disable its interrupt behavior.

MM2S Channel In Progress Register (0x00C)

This register gives the value of the Channel ID being serviced or worked upon. When MCDMA is running, this value is updated at run time based on the channel that is being serviced or worked upon. In idle condition, this register holds the last completed channel ID. It is possible for this value to have a mismatch with the TDEST on the AXI4-Stream port.

Table 2-10: **MM2S Channel In Progress Register (0x00C)**

Bits	Name	Default Value	Access	Description
31:16	Reserved	0	R	NA
15:0	Channel ID	0x0*	R	This is the channel ID that was last serviced. This register has a one-hot value to identify the channel that caused the error. A value of 1 corresponds to TDEST = 0, a value of 2 corresponds to TDEST = 1, a value of 4 corresponds to TDEST = 2 and so on.

Note: If the IP is configured to have Number of channels = 1, this value will always return 1.

MM2S Error Register (0x010)

This register gives the error status of the MM2S channel. This register is updated when MCDMA detects an error on any MM2S channel. All the bits in this register are read only (RO). Any error results in complete (all channels) halt of the MCDMA engine. The MCDMA has to be reset to clear the errors and the programming has to be done all over again (for all channels).

Table 2-11: **MM2S Error Register (0x010)**

Bits	Name	Default Value	Access	Description
31:7	RSVD			
6	SGDecErr	0	RO	Scatter Gather Decode Error. This error occurs if CURDESC_PTR and/or NXTDESC_PTR point to an invalid address. This error condition causes the AXI MCDMA to halt gracefully. The MCDMACR.RS bit is set to 0 and when the engine has completely shut down, the MCDMASR.Halted bit is set to 1. • 0 = No SG Decode Errors. • 1 = SG Decode Error detected. MCDMA Engine halts. This error cannot be logged into the descriptor.

Table 2-11: **MM2S Error Register (0x010) *(Cont'd)***

Bits	Name	Default Value	Access	Description
5	SGSlvErr	0	R0	Scatter Gather Slave Error. This error occurs if the slave read from on the Memory Map interface issues a Slave Error. This error condition causes the AXI MCDMA to gracefully halt. The MCDMACR.RS bit is set to 0, and when the engine has completely shut down, the MCDMASR.Halted bit is set to 1. • 0 = No SG Slave Errors. • 1 = SG Slave Error detected. MCDMA Engine halts. This error cannot be logged into the descriptor.
4	SGIntErr	0	RO	Scatter Gather Internal Error. This error occurs if a descriptor with the Complete bit already set is fetched. This indicates to the SG Engine that the descriptor is a tail descriptor. This error condition causes the AXI MCDMA to halt gracefully. The MCDMACR.RS bit is set to 0, and when the engine has completely shut down, the MCDMASR.Halted bit is set to 1. • 0 = No SG Internal Errors. • 1 = SG Internal Error detected. This error cannot be logged into the descriptor.
3	RSVD			
2	DMA Dec Err	0	RO	MCDMA Decode Error. This error occurs if the address request points to an invalid address. This error condition causes the AXI MCDMA to halt gracefully. The MCDMACR.RS bit is set to 0, and when the engine has completely shut down, the MCDMASR.Halted bit is set to 1. • 0 = No MCDMA Decode Errors. • 1 = MCDMA Decode Error detected. This bit can be set when such an event occurs on any of the channels.

Table 2-11: **MM2S Error Register (0x010) *(Cont'd)***

Bits	Name	Default Value	Access	Description
1	DMA SLv Err	0	RO	MCDMA Slave Error. This error occurs if the slave read from the Memory Map interface issues a Slave Error. This error condition causes the AXI MCDMA to halt gracefully. The MCDMACR.RS bit is set to 0 and when the engine has completely shut down the MCDMASR.Halted bit is set to 1. • 0 = No MCDMA Slave Errors. • 1 = MCDMA Slave Error detected. This bit can be set when such an event occurs on any of the channels.
0	DMA Intr Err	0	RO	MCDMA Internal Error. This error occurs if the buffer length specified in the fetched descriptor is set to 0. This error condition causes the AXI MCDMA to halt gracefully. The MCDMACR.RS bit is set to 0, and when the engine has completely shut down, the MCDMASR.Halted bit is set to 1. • 0 = No MCDMA Internal Errors. • 1 = MCDMA Internal Error detected. This bit is set when such an event occurs on any of the channels.

MM2S Channel Queue Scheduler Type (0x014)

This register specifies the handling of MM2S BD queues of channels. There can be three ways by which the BD queues can be handled. These can also be set at the time of IP configuration or can be changed at run time by way of register programming.

Table 2-12: **MM2S Channel Queue Scheduler Type (0x014)**

Bits	Name	Default Value	Access	Description
31:3	RSVD			
2	WRR Fair Distribution	0*	W/R, RO	When set to 1, the BDs are processed in the WRR Fair Distribution manner. This bit can be set only when the IP is configured to have programmable service type (c_mm2s_scheduler = 0). If the IP is configured for any other scheduler mechanism, this is RO bit.

Table 2-12: **MM2S Channel Queue Scheduler Type (0x014) *(Cont'd)***

Bits	Name	Default Value	Access	Description
1	WRR	1*	W/R, RO	When set to 1, the BDs will be processed in a WRR manner. This bit can be set only when the IP is configured to have a programmable service type (c_mm2s_scheduler = 0). If the IP is configured for any other scheduler mechanism, this is a RO bit.
0	Strict Priority	0*	W/R, RO	When set to 1, the BDs will be processed in a Strict Priority manner. This bit can be set only when the IP is configured to have a programmable service type (c_mm2s_scheduler = 0). If the IP is configured for any other scheduler mechanism, this is an RO bit.

The bits [2:0] can only have 100, 010 or 001 as valid values. Any other value will result in WRR scheduling. This register should be updated only when MM2S is idle and can be updated only when IP is configured with the scheduler option set to Programmable.

MM2S BD queue, for any channel, should end only with the EOF bit set. Ensure that the queue is populated correctly. In the absence of this, the MCDMA engine can stall infinitely or send incorrect data over the AXI4-Stream.

Note: The default value is based on the Scheduler scheme that is selected at IP configuration time. When the scheduler selected is Programmable, the default value is set to 010. When set to Strict Priority, default value is 001, when set to WRR-FD, it is 100.

MM2S Channel Weight Registers (0x018, 0x01C)

This register is used to specify the weights for each of the channels. Register 0x018 is used for the first eight channels, while register 0x01C is used for the remaining eight channels. Programming weights for disabled channels has no effect other than a change in value of the register. This register can be programmed at any time during the operation, however, the new values come into effect only when the scheduler iteration starts. See the MM2S Queue Scheduler for more information.

Table 2-13: **MM2S Channel Weight Registers (0x018)**

Bits	Name	Default Value	Access	Description
3:0	Weight for Ch1	1	R/W	Specifies the number of packets to be sent in one iteration.
7:4	Weight for Ch2	1	R/W	Specifies the number of packets to be sent in one iteration.
11:8	Weight for Ch3	1	R/W	Specifies the number of packets to be sent in one iteration.

Table 2-13: **MM2S Channel Weight Registers (0x018) *(Cont'd)***

Bits	Name	Default Value	Access	Description
15:12	Weight for Ch4	1	R/W	Specifies the number of packets to be sent in one iteration.
19:16	Weight for Ch5	1	R/W	Specifies the number of packets to be sent in one iteration.
23:20	Weight for Ch6	1	R/W	Specifies the number of packets to be sent in one iteration.
27:24	Weight for Ch7	1	R/W	Specifies the number of packets to be sent in one iteration.
31:28	Weight for Ch8	1	R/W	Specifies the number of packets to be sent in one iteration.

Table 2-14: **MM2S Channel Weight Registers (0x01C)**

Bits	Name	Default Value	Access	Description
3:0	Weight for Ch9	1	R/W	Specifies the number of packets to be sent in one iteration.
7:4	Weight for Ch10	1	R/W	Specifies the number of packets to be sent in one iteration.
11:8	Weight for Ch11	1	R/W	Specifies the number of packets to be sent in one iteration.
15:12	Weight for Ch12	1	R/W	Specifies the number of packets to be sent in one iteration.
19:16	Weight for Ch13	1	R/W	Specifies the number of packets to be sent in one iteration.
23:20	Weight for Ch14	1	R/W	Specifies the number of packets to be sent in one iteration.
27:24	Weight for Ch15	1	R/W	Specifies the number of packets to be sent in one iteration.
31:28	Weight for Ch16	1	R/W	Specifies the number of packets to be sent in one iteration.

MM2S Channels Completed Registers (0x020)

This register reports the channels that were serviced at a given instance of time. This register is helpful in identifying the channels on which packets were sent.

Table 2-15: **MM2S Channels Completed Registers (0x020)**

Bits	Name	Default Value	Access	Description
31:16	Reserved	0	R	NA
15:0	Channel ID	0x0	RC	This is the channel that was serviced. This register has a one-hot value to identify the channel that caused the error. A value of 1 corresponds to TDEST = 0, a value of 2 corresponds to TDEST = 1, a value of 4 corresponds to TDEST = 2 and so on. This register is cleared on read.

A channel is considered to be serviced when BD/BDs with start-of-frame (SOF) and EOF bits are processed and updated back in the memory.

MM2S ARCACHE and ARUSER Registers (0x024)

Program this register to set the desired value for `arcache` and `aruser` ports of the AXI4 read interface (MM2S Memory Map Interface). This register should not be programmed while the MCDMA is non idle.

Table 2-16: **MM2S ARCACHE and ARUSER Registers (0x024)**

Bits	Name	Default Value	Access	Description
31:16	Reserved	0	R	NA
15:12	Reserved	0	RW	Unused
11:8	Aruser value	0	RW	Program the desired aruser value.
7:4	Reserved	0	RW	Unused
3:0	Arcache value	0x3	RW	Program the desired arcache value.

MM2S Channel Interrupt Monitor Register (0x028)

This register gives the information about which channel(s) generated the Interrupt. This register can be used to identify the channel(s) that generated interrupts when all the Interrupt outputs of MCDMA are ORed at the system level. After the channel(s) are identified, make sure to read the Status register of the corresponding channel to get more information about the interrupt cause.

Table 2-17: **MM2S Channel Interrupt Monitor Register (0x028)**

Bits	Name	Default Value	Access	Description
31:16	Reserved	0	R	NA
15:0	Channel ID	0x0	R	This gives information about the channel(s) that generated the interrupt. This register has a one-hot value to identify the channel(s) that generated the interrupt. A value of 1 corresponds to Channel 0, a value of 2 corresponds to Channel 1, a value of 4 corresponds to Channel 2 and so on. Bits are automatically cleared when the corresponding interrupt is cleared in the channel status register.

MM2S Channel Observer Group 1-6 (0x440 - 454)

These registers provide information about channel(s) that were serviced based on the grouping of the channels done at IP customization. These registers are sub-sets of register 0x020. These registers are cleared on read.

Table 2-18: **MM2S Channel Observer Group 1-6 (0x440 - 454)**

Bits	Name	Default Value	Access	Description
31:16	Reserved	0	R	NA
15:0	Channel ID	0x0	RC	This is the channel that was serviced. This register has a one-hot value to identify the channel that was processed. A value of 1 corresponds to TDEST = 0, a value of 2 corresponds to TDEST = 1, a value of 4 corresponds to TDEST = 2 and so on. This register is cleared on read.

If at IP customization, Group 1 was allocated for Channels 0, 3 and 5, then register 0x440 will display the status only for those channels. If Channels 0, 1, 2, 3, 4 and 5 were serviced, this register will contain a value of 000000000101001.

This register is useful in multi-core (Observer) system where MCDMA is a shared resource for all cores. MCDMA IP supports maximum of six cores and 16 Channels can be distributed across each core as a static configuration. The Channel Observer register is available for each group and provides the status about the channels in a group being serviced.

Note: The following set of registers are present per Channel.

Channel Control Register

Each channel has its own Control register. This register provides control over an individual channel.

Table 2-19: **Channel Control Register**

Bits	Name	Default Value	Access	Description
31:24	IRQ DELAY	0	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 MCDMA engine to generate an interrupt after the delay time period has expired. The timer begins counting at the end of a packet and resets with the receipt of a new packet or a timeout event occurs. This counter operates on the clock that is connected to the Scatter Gather clock port. The delay counter is decremented once every 125 clocks. ***Note:*** Setting this value to zero disables the delay timer interrupt.
23:16	IRQ Threshold	0	R/W	Interrupt Threshold. This value is used for setting the interrupt threshold. When Interrupt on Complete (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 MCDMA engine. ***Note:*** The minimum setting for the threshold is 0x01. A write of 0x00 to this register has no effect.
15:8	RSVD			
7	Err Irq En	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.
6	DlyIrqEn	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.

Table 2-19: **Channel Control Register *(Cont'd)***

Bits	Name	Default Value	Access	Description
5	IOC_IrqEn	0	R/W	Interrupt on Complete Interrupt Enable. When set to 1, allows Interrupt On Complete events to generate an interrupt out for descriptors with the Complete bit set. • 0 = IOC Interrupt disabled. • 1 = IOC Interrupt enabled.
4	RSVD			
3	Err_on_other_IrqEn	0	R/W	Interrupt on Error on Other Channel Enable. When set to 1, allows Interrupt On Error on Other Channels events to generate an interrupt out. • 0 = Interrupt disabled. • 1 = Interrupt enabled.
2:1	RSVD			
0	CHANNEL.Fetch (CH.RS)	0	R/W	Start/Stop the fetch of BDs for this channel. When set to 1, this bit should not be set to 0. Setting this bit to 0 while the MCDMA is in run mode will result in undefined behavior.

Channel Status Register

Each channel has its own Status register. This register will provide the status of each channel.

Table 2-20: **Channel Status Register**

Bits	Name	Default Value	Access	Description
31:24	IRQ DELAY Status	0	R	Interrupt delay time Status. Indicates current interrupt delay time value.
23:16	IRQ Threshold Status	0	R	Interrupt Threshold Status. Indicates current interrupt threshold value.
15:8	RSVD			
7	Err Irq	0	R/WC	Interrupt on Error. When set to 1, indicates an interrupt event was generated on an error. If the corresponding bit in the Control register is enabled (Err_IrqEn = 1), an interrupt out is generated from the AXI MCDMA. Writing a 1 to this bit clears it.

Table 2-20: **Channel Status Register *(Cont'd)***

Bits	Name	Default Value	Access	Description
6	DlyIrq	0	R/WC	Interrupt on Delay. When set to 1, indicates an interrupt event was generated on delay timer timeout. If the corresponding bit in the Control register is enabled (Dly_IrqEn = 1), an interrupt out is generated from the AXI MCDMA. • 0 = No Delay Interrupt. • 1 = Delay Interrupt detected. Writing a 1 to this bit clears it.
5	IOC_Irq	0	R/WC	Interrupt on Complete. When set to 1, an interrupt event was generated on completion of a descriptor. This occurs for descriptors with the EOF bit set. If the corresponding bit in the Control register is enabled (IOC_IrqEn = 1) and if the interrupt threshold has been met, causes an interrupt out to be generated from the AXI MCDMA. • 0 = No IOC Interrupt. • 1 = IOC Interrupt detected. Writing a 1 to this bit clears it.
4	RSVD			
3	Err_on_other_Irq	0	R/WC	Interrupt on Error on other channels. When set to 1 an interrupt event was generated on detection of an error on other channels (MM2S or S2MM). If the corresponding bit in control register is enabled, it causes an interrupt out to be generated from AXI MCDMA. 0 = No Interrupt. 1 = Interrupt detected. Writing a 1 to this bit clears it.
2:1				
0	Idle	0	RO	MCDMA Channel Idle. Indicates the SG Engine has reached the tail pointer for the associated channel and there are no BDs queued. Writing to the tail pointer register automatically restarts the BD fetch operations. • 0 = Not Idle. • 1 = Idle.

Channel Current Descriptor Registers

Each channel has its own Current Descriptor (CD) register.

Table 2-21: **Channel Current Descriptor (CD) Registers**

Bits	Name	Default Value	Access	Description
31:6	Current descriptor address	0	R/W, RO	Specifies the address of the current descriptor. This value cannot be updated after the MCDMA is put in run mode. When CH.RS is 1, CURDESC_PTR becomes Read Only (RO). When the MCDMA Engine is running (MCDMACR.RS=1), the CURDESC_PTR register is updated by AXI MCDMA to indicate current descriptor that is being worked on. On error detection, CURDESC_PTR is updated to reflect the descriptor associated with the detected error.
5:0	RSVD	0	NA	

If the IP is configured for address width greater than 32 (that is, parameter c_addr_width > 32), specify the remaining MSB bits of CD in the MSB register as shown in Table 2-22.

Table 2-22: **Channel Current Descriptor (CD) Registers (Greater than 32 Bits)**

Bits	Name	Default Value	Access	Description
31:0	Current descriptor address (MSB)	0	R/W	Specifies the address of the current descriptor. This register is used only when Addr is > 32 bits. This value cannot be updated after the MCDMA is put in run mode.

Channel Tail Descriptor Register

Each channel has its own Tail Descriptor (TD) registers.

Table 2-23: **Channel Tail Descriptor (TD) Register**

Bits	Name	Default Value	Access	Description
31:6	Tail descriptor address	0	R/W	Specifies the address of the Tail descriptor. This value can be updated while the MCDMA is running. Writing into this register triggers the BD fetch. Indicates the pause pointer in a descriptor chain. The SG Engine pauses descriptor fetching after completing operations on the descriptor whose current descriptor pointer matches the tail descriptor pointer. A write by the CPU to the TAILDESC_PTR register causes the SG Engine to start fetching descriptors or restarts if it was idle (MCDMASR.Idle = 1). If it was not idle, writing to TAILDESC_PTR has no effect except to reposition the pause point. If the AXI MCDMA Channel MCDMACR.RS bit is set to 0, a write by the CPU to the TAILDESC_PTR register has no effect except to reposition the pause point.
5:0	RSVD	0	NA	

If the IP is configured for address width greater than 32, then specify the remaining MSB bits of TD in MSB register as follows:

Table 2-24: **Channel Tail Descriptor (TD) Register (Greater than 32 Bits)**

Bits	Name	Default Value	Access	Description
31:0	Tail descriptor address (MSB)	0	R/W	Specifies the address of the current descriptor. This register is used only when Addr is > 32 bits. Writing to this register triggers the BD fetch.

Packets Processed Statistics

This register reports the number of packets processed for the respective channel. This is a roll over counter of 16 bits.

Table 2-25: **Packets Processed Statistics**

Bits	Name	Default Value	Access	Description
31:0	RSVD	0	R	
15:0	Packet processed count	0	R	Reports the number of packets processed for the channel. This counter rolls over to 0 after reaching maximum value.

S2MM Common Control Register (0x500)

The S2MM path has a Control register to control the MCDMA engine. This register allows you to Reset or put the MCDMA in Run mode.

Table 2-26: **S2MM Common Control Register (0x500)**

Bits	Name	Default Value	Access	Description
31:3	RSVD			
2	Multichannel MCDMA Reset (MCDMA.RST)	0	R/W	Soft reset for resetting the AXI MCDMA core. Setting this bit to a 1 causes the AXI MCDMA to be reset. Reset is accomplished gracefully. Pending commands/transfers are flushed or completed. 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. This resets all the channels and clears all the registers. Re-configure the MCDMA after a soft reset. Resetting the S2MM also resets the MM2S.

Table 2-26: **S2MM Common Control Register (0x500) *(Cont'd)***

Bits	Name	Default Value	Access	Description
1	RSVD			
0	Multichannel MCDMA RunStop (MCDMA.RS)	0	R/W	Start/Stop the Multichannel S2MM MCDMA channel. • 0 = Stop – MCDMA stops when current (if any) MCDMA operations are complete. For Scatter/Gather Mode pending commands/transfers are flushed or completed. Descriptors in the update queue are allowed to finish updating to remote memory before engine halt. Data integrity on S2MM AXI4 cannot be guaranteed. The halted bit in the MCDMA Status register asserts to 1 when the MCDMA engine is halted. This bit is cleared by AXI MCDMA hardware when an error occurs. The CPU can also choose to clear this bit to stop MCDMA operations. Incoming packets are dropped when MCDMA is in stop mode. • 1 = Run – Start MCDMA operations. The halted bit in the MCDMA Status register deasserts to 0 when the MCDMA engine begins operations.

S2MM Common Status Register (0x504)

The S2MM path has a Common Status register for the tracking the MCDMA status. This register provides the overall status of the MCDMA engine.

Table 2-27: **S2MM Common Status Register (0x504)**

Bits	Name	Default Value	Access	Description
31:2	RSVD			
1	Idle	0	RO	MCDMA S2MM Idle. Indicates the state of AXI MCDMA operations. When IDLE, indicates the SG Engine has reached the tail pointer for all the channels and all queued descriptors have been processed. Writing to the tail pointer register to any channel automatically restarts MCDMA operations. • 0 = Not Idle. • 1 = Idle.
0	Halted (MCDMA.Halted)	1	RO	MCDMA Halted. Indicates the run/stop state of the MCDMA. • 0 = MCDMA channel running. • 1 = MCDMA.RS bit is set to 0. There can be a lag of time between when MCDMACR.RS = 0 and when MCDMASR.Halted = 1.

S2MM Channel Enable/Disable Register (0x508)

This register provides the capability to enable/disable a particular channel from being serviced.

Table 2-28: **S2MM Channel Enable/Disable Register (0x508)**

Bits	Name	Default Value	Access	Description
31:16	Reserved	0	R	NA
15:0	Channel Enable	0x1	R/W	Setting a bit to 1 enables the channel for service. Each bit corresponds to a channel. Bit [0] corresponds to a channel with TDEST = 0. Bit[1] corresponds to a channel with TDEST = 1 and so on.

If the MCDMA receives a packet on a disabled channel, the entire packet is dropped by the MCDMA engine. This register does not stop the BD fetch of the channel; it only stops a particular channel from being serviced. This register can be programmed at any time, but it will come into effect only when there is no data present on the S2MM AXI4-Stream interface or on arrival of the next packet.

Note: Disabling a channel does not disable its interrupt behavior.

S2MM Channel In Progress Register (0x50C)

This register gives the value of the Channel ID being serviced. When MCDMA is running, this value is updated at run time based on the channel that is being serviced. In idle condition, this register holds the value of the last channel that was serviced.

Table 2-29: **S2MM Channel In Progress Register (0x50C)**

Bits	Name	Default Value	Access	Description
31:16	Reserved	0	R	NA
15:0	Channel ID	0x0	R	This is the channel ID that was last serviced. This register has a one-hot value to identify the channel that caused the error. A value of 1 corresponds to TDEST = 0, a value of 2 corresponds to TDEST = 1, a value of 4 corresponds to TDEST = 2 and so on.

S2MM Error Register (0x510)

This register gives the error status of the MCDMA. This register is updated when MCDMA detects an error on any channel. All the bits in this register are RO. An error on any channel results in complete (all channels) halt of the MCDMA engine. The MCDMA has to be reset to clear the errors and the programming has to be done all over again (for all channels).

Table 2-30: **S2MM Error Register (0x510)**

Bits	Name	Default Value	Access	Description
31:7	RSVD			
6	SGDecErr	0	RO	Scatter Gather Decode Error. This error occurs if CURDESC_PTR and/or NXTDESC_PTR point to an invalid address. This error condition causes the AXI MCDMA to halt gracefully. The MCDMACR.RS bit is set to 0 and when the engine has completely shut down, the MCDMASR.Halted bit is set to 1. • 0 = No SG Decode Errors. • 1 = SG Decode Error detected. MCDMA Engine halts. This error cannot be logged into the descriptor.
5	SGSlvErr	0	R0	Scatter Gather Slave Error. This error occurs if the slave read from on the Memory Map interface issues a Slave Error. This error condition causes the AXI MCDMA to gracefully halt. The MCDMACR.RS bit is set to 0, and when the engine has completely shut down, the MCDMASR.Halted bit is set to 1. • 0 = No SG Slave Errors. • 1 = SG Slave Error detected. MCDMA Engine halts. This error cannot be logged into the descriptor.

Table 2-30: **S2MM Error Register (0x510) *(Cont'd)***

Bits	Name	Default Value	Access	Description
4	SGIntErr	0	RO	Scatter Gather Internal Error. This error occurs if a descriptor with the Complete bit already set is fetched. This indicates to the SG Engine that the descriptor is a tail descriptor. This error condition causes the AXI MCDMA to halt gracefully. The MCDMACR.RS bit is set to 0, and when the engine has completely shut down, the MCDMASR.Halted bit is set to 1. • 0 = No SG Internal Errors. • 1 = SG Internal Error detected. This error cannot be logged into the descriptor.
3	RSVD			
2	DMA Dec Err	0	RO	MCDMA Decode Error. This error occurs if the address request points to an invalid address. This error condition causes the AXI MCDMA to halt gracefully. The MCDMACR.RS bit is set to 0, and when the engine has completely shut down, the MCDMASR.Halted bit is set to 1. • 0 = No MCDMA Decode Errors. • 1 = MCDMA Decode Error detected. This bit can be set when such an event occurs on any of the channels.
1	DMA SLv Err	0	RO	MCDMA Slave Error. This error occurs if the slave read from the Memory Map interface issues a Slave Error. This error condition causes the AXI MCDMA to halt gracefully. The MCDMACR.RS bit is set to 0 and when the engine has completely shut down the MCDMASR.Halted bit is set to 1. • 0 = No MCDMA Slave Errors. • 1 = MCDMA Slave Error detected. This bit can be set when such an event occurs on any of the channels.
0	DMA Intr Err	0	RO	MCDMA Internal Error. This error occurs if the buffer length specified in the fetched descriptor is set to 0. Also, when in Scatter Gather Mode and using the status app length field, this error occurs when the Status AXI4-Stream packet RxLength field does not match the S2MM packet being received by the S_AXIS_S2MM interface. This error condition causes the AXI MCDMA to halt gracefully. The MCDMACR.RS bit is set to 0, and when the engine has completely shut down, the MCDMASR.Halted bit is set to 1. • 0 = No MCDMA Internal Errors. • 1 = MCDMA Internal Error detected. This bit is set when such an event occurs on any of the channels.

S2MM Packet Drop Status (0x514)

This register reports the number of packets dropped across all channels. This register is incremented when the MCDMA is put in run mode.

Table 2-31: **S2MM Packet drop Status (0x514)**

Bits	Name	Default Value	Access	Description
31:0	Packet drop count	0	R	Reports the number of packets dropped across all channels. This value increments by 1 every time a packet is dropped on any channel. The counter wraps around after it has reached maximum value.

S2MM Channels Completed Register (0x518)

This register reports the channels that were serviced at a given instance of time. This register is helpful in identifying the channels that received the packets.

Table 2-32: **S2MM Channels Completed Register (0x518)**

Bits	Name	Default Value	Access	Description
31:16	Reserved	0	R	NA
15:0	Channel ID	0	RC	This is the channel that was serviced. This register has a one-hot value to identify the channel that was processed. A value of 1 corresponds to TDEST = 0, a value of 2 corresponds to TDEST = 1, a value of 4 corresponds to TDEST = 2 and so on. This register is cleared on read.

For example, if MCDMA receives and processes packets on CH1, CH2, CH4 and drops all the packets on CH3, this register has a value of 0000000000001011 set.

S2MM AWCACHE and AWUSER Register (0x51C)

This register is used to program the value that should be driven on `awuser` and `awcache` ports of the AXI4 write interface (S2MM MMap Interface). This register should not be modified while the MCDMA is operating.

Table 2-33: **S2MM AWCACHE and AWUSER Register (0x51C)**

Bits	Name	Default Value	Access	Description
31:16	Reserved	0	R	NA
15:12	Reserved	0	RW	Unused
11:8	Awuser value	0	RW	Program the desired awuser value.
7:4	Reserved	0	RW	Unused
3:0	Awcache value	0x3	RW	Program the desired awcache value.

S2MM Channel Interrupt Monitor Register (0x520)

This register gives information about which channels generated the interrupt. This register can be used to identify the channel(s) that generated an interrupt when all the interrupt outputs of MCDMA are ORed at the system level. After the channels are identified, you should read the Status register of the corresponding channel to get more information about the interrupt cause.

Table 2-34: **S2MM Channel Interrupt Monitor Register (0x520)**

Bits	Name	Default Value	Access	Description
31:16	Reserved	0	R	NA
15:0	Channel ID	0	R	This gives info about the channel(s) that generated the interrupt. This register has a one-hot value to identify the channel(s) that generated the interrupt. A value of 1 corresponds to Channel 0, a value of 2 corresponds to Channel 1, a value of 4 corresponds to Channel 2 and so on. Bits are automatically cleared when the corresponding interrupt is cleared in the channel status register.

S2MM Channel Observer Group 1-6 Registers (0x940 – 0x954)

These registers give the information about which channel(s) were serviced based on the grouping of the channels done (at IP configuration). These registers are subsets of register 0x518. These registers are cleared on read.

Table 2-35: **S2MM Channel Observer Group 1-6 Registers (0x940 – 0x954)**

Bits	Name	Default Value	Access	Description
31:16	Reserved	0	R	NA
15:0	Channel ID	0	RC	This is the channel that was serviced. This register has a one-hot value to identify the channel that was processed. A value of 1 corresponds to TDEST = 0, a value of 2 corresponds to TDEST = 1, a value of 4 corresponds to TDEST = 2 and so on. This register is cleared on read.

If at IP customization, Group 1 was allocated for Channels 0, 3 and 5, then register 0x940 will display the status only for those channels. If Channels 0, 1, 2, 3, 4 and 5 were serviced, this register contains a value of 000000000101001.

This register is useful in a multi-core (Observer) system where MCDMA is a shared resource for all cores. MCDMA IP support a maximum of 6 cores and 16 Channels can be distributed across each core as a static configuration. The Channel Observer register is available for each group and provides the status about the channels in a group being serviced.

Note: The following set of registers are present per Channel.

Channel Control Register

Each channel has its own Control register. This register provides control over individual channels.

Table 2-36: **Channel Control Register**

Bits	Name	Default Value	Access	Description
31:24	IRQ DELAY	0	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 MCDMA engine to generate an interrupt after the delay time period has expired. The timer begins counting at the end of a packet and resets with the receipt of a new packet or a timeout event occurs. This counter operates on the clock that is connected to the Scatter Gather clock port. The delay counter is decremented once every 125 clocks. ***Note:*** Setting this value to zero disables the delay timer interrupt.
23:16	IRQ Threshold	0x1	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 MCDMA engine. ***Note:*** The minimum setting for the threshold is 0x01. A write of 0x00 to this register has no effect.
15:8	IRQ Packet Drop Threshold	0x1	R/W	Packet Drop Interrupt Threshold. This value is used for setting the interrupt threshold for a packet drop. When a packet is dropped on a channel, an internal counter counts down from this threshold setting. When the count reaches zero, an interrupt out is generated by the MCDMA engine. This value is decremented for every packet drop event occurring on a channel. ***Note:*** The minimum setting for the threshold is 0x01. A write of 0x00 to this register has no effect.
7	Err Irq En	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.

Table 2-36: **Channel Control Register *(Cont'd)***

Bits	Name	Default Value	Access	Description
6	DlyIrqEn	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.
5	IOC_IrqEn	0	R/W	Interrupt on Complete Interrupt Enable. When set to 1, allows Interrupt On Complete events to generate an interrupt out for descriptors with the Complete bit set. • 0 = IOC Interrupt disabled. • 1 = IOC Interrupt enabled.
4	Pktdrop_IrqEn	0	R/W	Interrupt on Packet drop enable. When set to 1, allows an Interrupt on Packet drop for a channel.
3	Err_on_other En	0	R/W	Interrupt on Error on other channels Interrupt Enable. When set to 1, allows error events on other channels to generate an interrupt out. • 0 = Interrupt disabled. • 1 = Interrupt enabled.
2:1	RSVD	0		
0	CHANNEL. RunStop (CH.RS)	0	R/W	Start/Stop the fetch of BDs for this channel. After set to 1, this bit should not be set to 0. Setting this bit to 0 while the MCDMA is in run mode will result in undefined behavior.

Channel Status Register

Each channel has its own Status register. This register provides the status of each channel.

Table 2-37: **Channel Status Register**

Bits	Name	Default Value	Access	Description
31:24	IRQ DELAY Status	0	R	Interrupt delay time Status. Indicates current interrupt delay time value.
23:16	IRQ Threshold Status	0x1	R	Interrupt Threshold Status. Indicates current interrupt threshold value.
15:8	IRQ Packet Drop Status	0x1	R	Interrupt Packet Drop Threshold. Indicates current interrupt threshold value.

Table 2-37: **Channel Status Register *(Cont'd)***

Bits	Name	Default Value	Access	Description
7	Err Irq	0	R/WC	Interrupt on Error. When set to 1, indicates an interrupt event was generated on an error. If the corresponding bit in Control register is enabled (Err_IrqEn = 1), an interrupt out is generated from the AXI MCDMA. Writing a 1 to this bit clears it.
6	DlyIrq	0	R/WC	Interrupt on Delay. When set to 1, indicates an interrupt event was generated on delay timer timeout. If the corresponding bit in Control register is enabled (Dly_IrqEn = 1), an interrupt out is generated from the AXI MCDMA. • 0 = No Delay Interrupt. • 1 = Delay Interrupt detected. Writing a 1 to this bit clears it.
5	IOC_Irq	0	R/WC	Interrupt on Complete. When set to 1 an interrupt event was generated on completion of a descriptor. This occurs for descriptors with the EOF bit set. If the corresponding bit in the Control register is enabled (IOC_IrqEn = 1) and if the interrupt threshold has been met, causes an interrupt out to be generated from the AXI MCDMA. • 0 = No IOC Interrupt. • 1 = IOC Interrupt detected. Writing a 1 to this bit clears it.
4	Pktdrop_irq	0	R/WC	Interrupt on packet drop. When set to 1 indicates an interrupt event was generated on packet drop. If the corresponding bit in the Control register is enabled (PktDrp_IrqEn = 1) and if the interrupt threshold has been met, causes an interrupt out to be generated from the AXI MCDMA. • 0 = No packet drop Interrupt. • 1 = Packet drop Interrupt detected. Writing a 1 to this bit clears it.
3	Err_on_other_ch_irq	0	R/WC	Interrupt on Error on other channels. When set to 1, indicates an interrupt event was generated on an error on other channels. If the corresponding bit in Control register is enabled (Err_on_other En = 1), an interrupt out is generated from the AXI MCDMA. Writing a 1 to this bit clears it.

Table 2-37: **Channel Status Register *(Cont'd)***

Bits	Name	Default Value	Access	Description
2	RSVD			
1	BD ShortFall	0	RO	This bit is set when a packet is being processed and the BD queue becomes empty. This means that the packet that is being serviced is too large to be accommodated in the BD queue. This scenario leads to MCDMA waiting forever for the packet to get completed. To get over this, extend the BD chain and program the TD to fetch more BDs. This does not result in a packet drop. This bit can also get set momentarily when the MCDMA is servicing the last BD for a channel and accommodates the packet. After the TLAST is accommodated in the last BD, this bit is unset.
0	Idle (Queue Empty)	1	RO	MCDMA Channel Idle. Indicates the SG Engine has reached the tail pointer for the associated channel and all queued descriptors have been processed. This means that the BD queue is empty and there are no more BDs to process. Writing to the tail pointer register automatically restarts the BD fetch operations. • 0 = BD queue not empty. • 1 = BD Queue empty. If the packet arrives while this bit is set, that packet is dropped.

Channel Current Descriptor Registers

Each channel has its own Current Descriptor registers.

Table 2-38: **Channel Current Descriptor (CD) Registers**

Bits	Name	Default Value	Access	Description
31:6	Current descriptor address	0	R/W, RO	Specifies the address of the current descriptor. This value cannot be updated once the Channel is put in run mode. When CH.RS is 1, CURDESC_PTR becomes Read Only (RO). When the MCDMA Engine is running (MCDMA.RS=1), CURDESC_PTR register is updated by AXI MCDMA to indicate current descriptor that is being worked on. On error detection, CURDESC_PTR is updated to reflect the descriptor associated with the detected error
5:0	RSVD	0	NA	

If the IP is configured for address width greater than 32, specify the remaining MSB bits of CD in the MSB register as follows:

Table 2-39: **Channel Current Descriptor (CD) Registers — greater than 32 bits**

Bits	Name	Default Value	Access	Description
31:0	Current descriptor address (MSB)	0	R/W	Specifies the address of the current descriptor. This register is used only when Addr is > 32 bits. This value cannot be updated after the MCDMA is put in run mode.

Channel Tail Descriptor Register

Each channel has its own Tail Descriptor registers.

Table 2-40: **Channel Tail Descriptor (TD) Register**

Bits	Name	Default Value	Access	Description
31:6	Tail descriptor address	0	R/W	Specifies the address of the Tail descriptor. This value can be updated while the MCDMA is running. Writing into this register triggers the BD fetch. Indicates the pause pointer in a descriptor chain. The SG Engine pauses descriptor fetching after completing operations on the descriptor whose current descriptor pointer matches the tail descriptor pointer. When AXI MCDMA Channel is put in run mode (CH.RS = 1), a write by the CPU to the TAILDESC_PTR register causes the SG Engine to start fetching descriptors or restart if it was idle (CH = 1). If it was not idle, writing to TAILDESC_PTR has no effect except to reposition the pause point. If the AXI MCDMA Channel CH.RS bit is set to 0, a write by the CPU to the TAILDESC_PTR register has no effect except to reposition the pause point.
5:0	RSVD	0	NA	

If the IP is configured for an address width greater than 32, specify the remaining MSB bits of TD in MSB register as follows:

Table 2-41: **Channel Tail Descriptor (TD) Register — greater than 32 bits**

Bits	Name	Default Value	Access	Description
31:0	Tail descriptor address (MSB)	0	R/W	Specifies the address of the current descriptor. This register is used only when Addr is > 32 bits. BD fetch is triggered when this register is updated.

Packet Drop Statistic

This register reports the number of packets dropped across the channel.

Table 2-42: **Packet Drop Statistic**

Bits	Name	Default Value	Access	Description
31:0	RSVD	0	R	
15:0	Packet drop count	0	RC	Reports the number of packets dropped for the channel. This value increments by 1 every time a packet is dropped. The counter wraps around after it has reached maximum value. This register is cleared when read.

Packets Processed Statistics

This register reports the number of packets processed for the respective channel. This is a roll over counter of 16 bits.

Table 2-43: **Packets Processed Statistics**

Bits	Name	Default Value	Access	Description
31:0	RSVD	0	R	
15:0	Packet processed count	0	R	Reports the number of packets processed for the channel. This counter rolls over to 0 after reaching maximum value.

SG Cache/User Register (0x4B0)

Program this register to set the desired value for the `arcache`, `aruser`, `awcache` and `awuser` ports of the SG AXI4 interface. This register should not be programmed while the MCDMA is in operation.

Table 2-44: **SG Cache/User Register (0x4B0)**

Bits	Name	Default Value	Access	Description
31:28	Reserved	0	RW	Unused.
27:24	Awuser value	0	RW	Program the desired awuser value.
23:20	Reserved	0	RW	Unused.
19:16	Awcache value	0x3	RW	Program the desired awcache value.
15:12	Reserved	0	RW	Unused.
11:8	Aruser value	0	RW	Program the desired aruser value.
7:4	Reserved	0	RW	Unused
3:0	Arcache value	0x3	RW	Program the desired arcache value.

Scatter Gather Buffer Descriptor

This section defines the fields of the S2MM (Receive) and MM2S (Transmit) Scatter Gather Descriptors for AXI MCDMA. 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. The Buffer Length can describe up to 67,108,864 bytes of data buffer per descriptor. Each channel should maintain a separate BD chain for MM2S and S2MM.

Table 2-45: **MM2S BD Field Description**

Address Offset	Name	Description
00h	Next Descriptor	Next descriptor pointer
04h	Next Descriptor (MSB)	MSB (32 bits) next descriptor pointer
08h	Buffer Address	Buffer descriptor address
0Ch	Buffer Address (MSB)	MSB (32bits) buffer descriptor address
10h	RSVD	Reserved
14h	Control	Control Information for BD
18h	Control Sideband	Control Information for AXI4-Stream Sideband
1Ch	Status	Status field
20h	APP0	User application field 0
24h	APP1	User application field 1
28h	APP2	User application field 2
2Ch	APP3	User application field 3
30h	APP4	User application field 4

Table 2-46: **S2MM BD Field Description**

Address Offset	Name	Description
00h	Next Descriptor	Next descriptor pointer
04h	Next Descriptor (MSB)	MSB (32 bits) next descriptor pointer
08h	Buffer Address	Buffer descriptor address
0Ch	Buffer Address (MSB)	MSB (32bits) buffer descriptor address
10h	RSVD	Reserved
14h	Control	Control Information field
18h	Status	Status field

Table 2-46: **S2MM BD Field Description *(Cont'd)***

Address Offset	Name	Description
1Ch	Sideband Status	Status of sideband signals
20h	APP0	User application field 0
24h	APP1	User application field 1
28h	APP2	User application field 2
2Ch	APP3	User application field 3
30h	APP4	User application field 4

Notes:

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

The Next Descriptor and Buffer Address fields in MM2S and S2MM are identical and are described as follows:

Next Descriptor Pointer (0x00h). This value provides the pointer to the next descriptor in the chain.

Table 2-47: **Next Descriptor Pointer (0x00h)**

Bits	Name	Description
31: 6	Next 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.
5:0	RSVD	Reserved

Next Descriptor MSB Pointer (0x04h). This value provides the upper 32 bits of the pointer to the next descriptor in the descriptor chain. This is used only when AXI MCDMA is configured for an address space greater than 32. The fields are described as follows:

Table 2-48: **Next Descriptor MSB Pointer (0x04h)**

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

Buffer Address (0x08h). This value provides the pointer to the buffer space available to transfer data from stream to system memory. The fields are described as follows:

Table 2-49: **Buffer Address (0x08h)**

Bits	Name	Description
31: 0	Buffer Address	Provides the location of the buffer space available to store data transferred from stream to memory map. ***Note:*** If the Data Realignment Engine is included, the Buffer Address can be at any byte offset. If the Data Realignment Engine is not included the Buffer Address must be memory-mapped data width aligned.

MSB Buffer Address MSB (0x0Ch). This value provides the upper 32 bits of the pointer to the buffer space available to transfer data from stream to system memory. This is used only when AXI MCDMA is configured for an address space greater than 32. The fields are described as follows:

Table 2-50: **MSB Buffer Address MSB (0x0Ch)**

Bits	Name	Description
31: 0	Buffer Address (MSB)	Provides the MSB 32 bits of the location of the buffer space available to store data transferred from stream to memory map.

MM2S Control, Sideband, Status and APP fields

Control (0x14h). This value provides control for MM2S transfers from stream to memory map. The fields are described as follows:

Table 2-51: **Control (0x14h)**

Bits	Name	Description
31	TX SoF	Start of Frame. Flag indicating the first buffer to be processed. This flag is set by the CPU to indicate to AXI MCDMA 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.
30	TX EoF	End of Frame. Flag indicating the last buffer to be processed. This flag is set by the CPU to indicate to AXI MCDMA that this descriptor describes the end of the packet. The buffer associated with this descriptor is transmitted last and results in TLAST assertion. • 0 = Not End of Frame. • 1 = End of Frame.
29:26	Reserved	
25:0	Buffer Length	Indicates the size in bytes of the transfer buffer. This value indicates the amount of bytes to transmit out on the MM2S stream. The usable width of buffer length is specified by the parameter Width of Buffer Length Register (c_sg_length_width). A maximum of 67,108,864 bytes of transfer can be described by this field. This value should be an integer multiple of the AXI4-Stream data width; however it can have any value if the TX EOF bit is set.

Control Sideband (0x18h). The new field that contains the value of TID and TUSER that are presented on the last data beat of the AXI4-Stream data. The Sideband Status fields are described as follows:

Table 2-52: **Control Sideband (0x18h)**

Bits	Name	Description
31: 24	TID	This field contains the value of TID to be presented on the last data beat of the packet that is, the beat that has TLAST.
23:16	RSVD	Reserved.
15:0	TUSER	This field contains the value of TUSER to be presented on the last data beat of the packet that is, the beat that has TLAST.

Each channel has its own queue. Whenever CH1 is active, the TDEST presented on AXI4-Stream would be 0. TDEST is 1 when CH2 is active and so on.

Status (0x1Ch). This value provides the status for MM2S transfers from memory map to stream. This field is updated by the MCDMA.

Table 2-53: **Status (0x1Ch)**

Bits	Name	Description
31	Completed	Completed. This indicates to the software that the MCDMA engine has completed the transfer as described by the associated descriptor. The MCDMA Engine sets this bit to 1 when the transfer is completed. The software can manipulate any descriptor with the Completed bit set to 1. • 0 = Descriptor not completed. • 1 = Descriptor completed. ***Note:*** If a descriptor is fetched with this bit set to 1, the descriptor is considered a stale descriptor. A SGIntErr is flagged and the AXI MCDMA engine halts.
30	DMA DecErr	DMA Decode Error. Decode Error detected by primary AXI DataMover. This error occurs if the Descriptor Buffer Address points to an invalid address. This error condition causes the AXI MCDMA to halt gracefully. The MCDMACR.RS bit is set to 0, and when the engine has completely shut down, the MCDMASR.Halted bit is set to 1. • 0 = No MCDMA Decode Errors. • 1 = MCDMA Decode Error detected. MCDMA Engine halts.
29	DMA SlvErr	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 MCDMA to halt gracefully. The MCDMACR.RS bit is set to 0, and when the engine has completely shut down, the MCDMASR.Halted bit is set to 1. • 0 = No MCDMA Slave Errors. • 1 = MCDMA Slave Error detected. MCDMA Engine halts.
28	DMA IntErr	MCDMA Internal Error. Internal Error detected by 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 there is an under-run or over-run condition. This error condition causes the AXI MCDMA to halt gracefully. The MCDMACR.RS bit is set to 0, and when the engine has completely shut down, the MCDMASR.Halted bit is set to 1. • 0 = No MCDMA Internal Errors. • 1 = MCDMA Internal Error detected. MCDMA Engine halts.
27:26	Reserved	
25:0	Transferred Bytes	Indicates the size in bytes of the actual data transferred for this descriptor. This value indicates the amount of bytes to transmit out on MM2S stream. This value should match the Control Buffer Length field. The usable width of Transferred Bytes is specified by the parameter Width of Buffer. Length register (c_sg_length_width). A maximum of 67,108,864 bytes of transfer can be described by this field.

APP0 –APP4 Fields (0x20 – 0x30). This value provides the User Application field space for the MM2S to be sent over the Control Stream.

Table 2-54: **APP0 –APP4 Fields (0x20 – 0x30)**

Bits	Name	Description
31: 0	APP0-APP4	User application fields 0 to 4. Specifies user-specific application data. When Status Control Stream is enabled, the Application (APP) fields of the Start of Frame (SOF) Descriptor are transmitted to the AXI Control Stream. For other MM2S descriptors with SOF = 0, the APP fields are fetched but ignored.

S2MM Control, Status, Sideband and APP Fields

Control (0x14h). This value provides control for S2MM transfers from stream to memory map. The fields are described as follows:

Table 2-55: **Control (0x14h)**

Bits	Name	Description
31: 26	Reserved	
25:0	Buffer Length	This value indicates the amount of space in bytes available for receiving data in anS2MM stream. The usable width of buffer length is specified by the parameter Width of Buffer Length Register (c_sg_length_width). A maximum of 67,108,864 bytes of transfer can be described by this field. This value should be an integer multiple of AXI4-Stream data width. ***Note:*** The total buffer space in the S2MM descriptor chain (that is, the sum of buffer length values for each descriptor in a chain) must be, at a minimum, capable of holding the maximum receive packet size. Undefined results occur if a packet larger than the defined buffer space is received. ***Note:*** Setting the Buffer Length Register Width smaller than 26 reduces FPGA resource utilization.

Status (0x18h). This value provides the status for S2MM transfers from stream to memory map. This field is updated by the MCDMA. The fields are described as follows:

Table 2-56: **Status (0x18h)**

Bits	Name	Description
31	Completed	Completed. This indicates to the software that the MCDMA Engine has completed the transfer as described by the associated descriptor. The MCDMA Engine sets this bit to 1 when the transfer is completed. The software can manipulate any descriptor with the Completed bit set to 1. • 0 = Descriptor not completed. • 1 = Descriptor completed. ***Note:*** If a descriptor is fetched with this bit set to 1, the descriptor is considered a stale descriptor. A SGIntErr is flagged and the AXI MCDMA engine halts.
30	DMA DecErr	DMA Decode Error. Decode Error detected by primary AXI DataMover. This error occurs if the Descriptor Buffer Address points to an invalid address. This error condition causes the AXI MCDMA to halt gracefully. The MCDMACR.RS bit is set to 0, and when the engine has completely shut down, the MCDMASR.Halted bit is set to 1. • 0 = No MCDMA Decode Errors. • 1 = MCDMA Decode Error detected. MCDMA Engine halts
29	DMA SlvErr	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 MCDMA to halt gracefully. The MCDMACR.RS bit is set to 0, and when the engine has completely shut down, the MCDMASR.Halted bit is set to 1. • 0 = No MCDMA Slave Errors. • 1 = MCDMA Slave Error detected. MCDMA Engine halts.
28	DMA IntErr	MCDMA Internal Error. Internal Error detected by 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. This error condition causes the AXI MCDMA to halt gracefully. The MCDMACR.RS bit is set to 0, and when the engine has completely shut down, the MCDMASR.Halted bit is set to 1. • 0 = No MCDMA Internal Errors. • 1 = MCDMA Internal Error detected. MCDMA Engine halts.
27	RXSOF	Start of Frame. Flag indicating buffer holds first part of packet. This bit is set by AXI MCDMA to indicate that the buffer associated with this descriptor contains the start of the packet. • 0 = Not start of frame. • 1 = Start of frame.

Table 2-56: **Status (0x18h) *(Cont'd)***

Bits	Name	Description
26	RXEOF	End of Frame. Flag indicating buffer holds the last part of packet. This bit is set by AXI MCDMA to indicate that the buffer associated with this descriptor contains the end of the packet. • 0 = Not End of Frame. • 1 = End of Frame. ***Note:*** User Application data sent through the status stream input is stored in APP0 to APP4 of the RXEOF descriptor when the Control/Status Stream is enabled.
25:0	Transferred Bytes	This value indicates the amount of data received and stored in the buffer described by this descriptor. This might or might not match the buffer length. For example, if this descriptor indicates a buffer length of 1,024 bytes but only 50 bytes were received and stored in the buffer, then the Transferred Bytes field indicates 0x32. The entire receive packet length can be determined by adding the Transferred Byte values from each descriptor from the RXSOF descriptor to the Receive End of Frame (RXEOF) descriptor. ***Note:*** The usable width of Transferred Bytes is specified by the parameter Width of Buffer Length Register (c_sg_length_width). A maximum of 67,108,864 bytes of transfer can be described by this field. ***Note:*** Setting the Buffer Length Register Width smaller than 26 reduces FPGA resource utilization.

Sideband Status (0x1Ch). The new field contains the value of TID, TDEST and TUSER on the last data beat of the incoming AXI4-Stream data. This field is updated by the MCDMA. The Sideband Status fields are described as follows:

Table 2-57: **Sideband Status (0x1Ch)**

Bits	Name	Description
31: 24	TID	This field contains the value of TID present on the last data beat of the packet that is, the beat that has TLAST.
23:20	RSVD	Reserved.
19:16	TDEST	This field contains the value of TDEST.
15:0	TUSER	This field contains the value of TUSER present on the last data beat of the packet that is, the beat that has TLAST.

APP0 –APP3 Fields (0x20 – 0x2C). This value provides the User Application field space for the S2MM received status on the Status Stream.

Table 2-58: **APP0 –APP3 Fields (0x20 – 0x2C)**

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

APP4 Field (0x30). This value provides the User Application 4 field space for S2MM received status on the Status Stream.

Table 2-59: **APP4 Field (0x30)**

Bits	Name	Description
31: 0	APP4	User Application field 4 and Receive Byte Length. If Use RxLength In Status Stream is not enabled, this field functions identically to APP0 to APP3 in that the status data received on the AXI Status Stream is stored into the APP4 field of the End of Frame (EOF) Descriptor. This field has a dual purpose when Use RxLength in Status Stream is enabled. The first least significant bits specified in the Buffer Length Register Width (up to a maximum of 16) specify the total number of receive bytes for a packet that were received on the S2MM primary data stream. Second, the remaining most significant bits are User Application data.

Descriptor Management

Prior to starting MCDMA operations, the software application must set up a descriptor chain (one for each channel for MM2S and S2MM). When the AXI MCDMA begins processing the descriptors, it fetches, processes, and then updates the descriptors. A single Scatter Gather engine fetches the descriptors of all channels in a round robin manner while alternating between MM2S and S2MM descriptors.

By analyzing the descriptors, the software application can read the status on the associated MCDMA 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 MCDMA use. To prevent software and hardware from stepping on each other, a Tail Pointer Mode is created. The tail pointer is initialized by software to point to the end of the descriptor chain. This becomes the pause point for hardware. When hardware begins running, it fetches and processes each descriptor in the chain until it reaches the tail pointer.

The AXI MCDMA then pauses descriptor processing. The software is allowed to process and re-allocate any descriptor whose Complete bit is set to 1. The act of writing to the TAILDESC register causes the AXI MCDMA hardware, if it is paused at the tail pointer, to begin processing descriptors again. If the AXI MCDMA hardware is not paused at the TAILDESC pointer, writing to the TAILDESC register has no effect on the hardware. In this situation, the AXI MCDMA continues to process descriptors until reaching the new tail descriptor pointer location. Descriptor Management must be done by the software. AXI MCDMA does not manage the descriptors.

MM2S Descriptor Setting and AXIS Control Stream

The relationship between descriptor SOF/EOF settings and the AXI Control Stream is illustrated in Figure 2-2. The descriptor with SOF=1 is the beginning of the packet. The User Application fields for this descriptor are also presented on the AXI Control Stream if the Status/Control Stream is enabled. User Application fields following a descriptor with SOF=1, up to and including the descriptor with EOF =1, are ignored by the AXI MCDMA engine. If Status/Control Stream is disabled, the User Application fields are not fetched by the SG Fetch Engine.

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 MCDMA prior to, during, or after the primary data packet. Throttling by the target device is allowed, and throttling by AXI MCDMA can occur. AXI MCDMA 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. The MCDMA outputs the Control Stream for a channel before the start of the data phase.

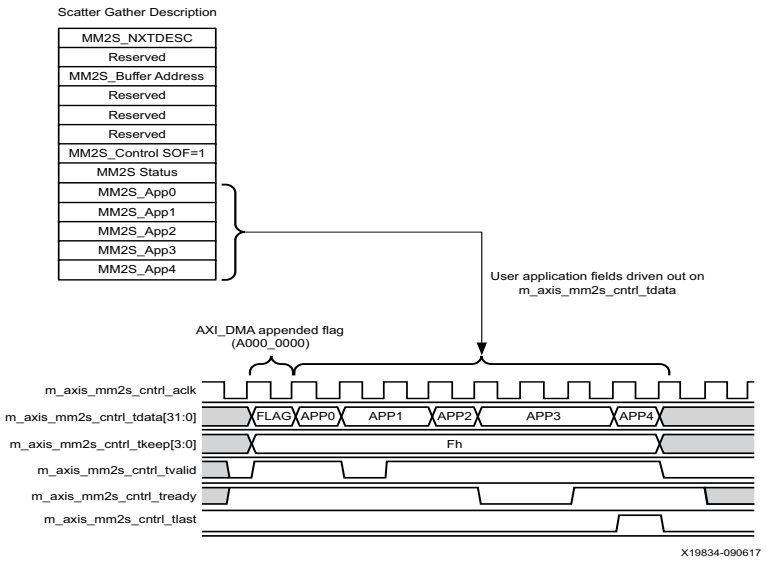

Figure 2-2: **Relationship Between Descriptor SOF/EOF Settings and the AXI Control Stream**

S2MM Descriptor Setting and AXI Status Stream

The relationship between the descriptor RXSOF/RXEOF settings and the AXI Status Stream is illustrated in Figure 2-3. The descriptor with RXSOF=1 describes the buffer containing the first part of the receive packet. The Descriptor with RXEOF=1 describes the buffer containing the last part of the receive packet. For proper operation, the software must specify enough buffer space (the sum of the buffer lengths in each descriptor of the descriptor chain) to be greater than or equal to the maximum sized packet that is received. If the Status/Control Stream is included, the status received is stored in the User Application fields (APP0 to APP4) of the descriptor with RXEOF set. The actual byte count of received and stored data for a particular buffer is updated to the Transferred Bytes field in the associated descriptor. The software can determine how many bytes were received by walking the descriptors from RXSOF to RXEOF and adding the Bytes Transferred fields to get a total byte count. For applications where you provide the total length in the status stream, this value is stored in the user-defined application location in the descriptor with RXEOF=1. The AXI status stream is provided for transfer of the 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 2-3, the status packet updates to the app fields of the detected last descriptor (RXEOF = 1) describing the packet. Normally, the status stream should come at the start of the S2MM data stream. If the Use RxLength In Status Stream is disabled, the status stream can come at any time during the course of the S2MM frame. The End of Frame (EOF) Buffer Descriptor (BD) update will happen only when the entire status stream is received.

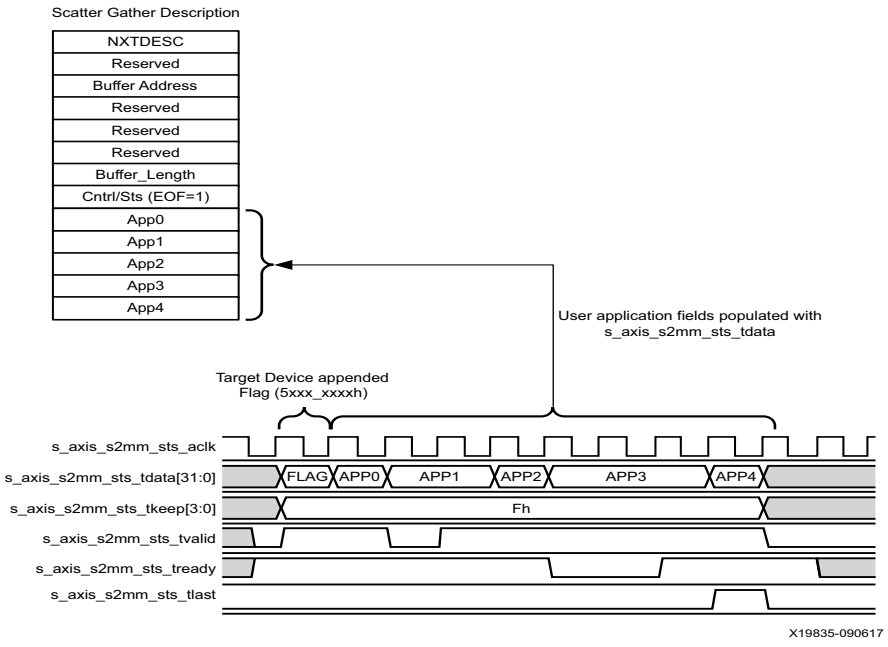

Figure 2-3: **Relationship Between Descriptor RXSOF/RXEOF Settings and the AXI Status Stream**

Designing with the Core

Typical System Comprising AXI MCDMA

The AXI MCDMA is useful in systems where multiple instances of AXI MCDMA are not a viable option. A single instance of AXI MCDMA is capable of processing descriptors and routing packet for up to 16 channels. The system microprocessor has access to the AXI MCDMA through the AXI4-Lite interface. An integrated Scatter/Gather engine fetches buffer descriptors from system memory which then coordinates primary data transfers between the AXI IP and DDRx. The multi-interrupt output of the AXI MCDMA core is routed to the System Interrupt Controller.

A typical use case is shown in Figure 3-1.

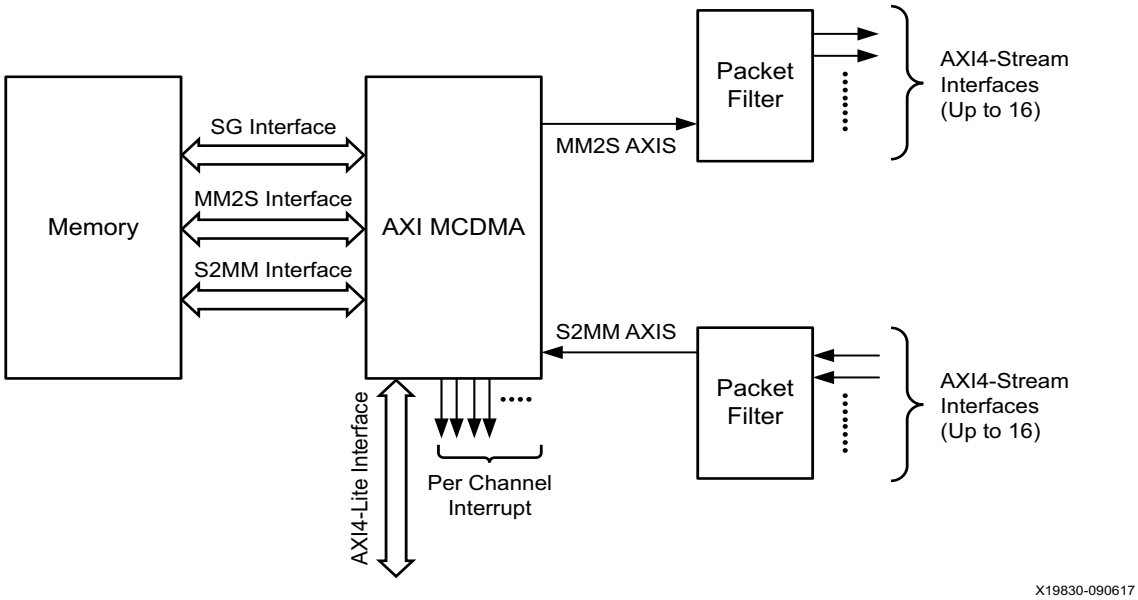

Figure 3-1: **Typical Use Case**

The Packet Filter module, which lies outside the AXI MCDMA and is a user-implemented logic, handles the streaming data. On the S2MM side, this module is responsible for putting a required TDEST value on the S2MM AXI4-Stream data. It is up to you determine and add the TDEST value. Similarly on the MM2S side, this module is responsible for routing the AXI4-Stream data based on the TDEST value present on the MM2S AXI4-Stream interface.

Clocking

There are five possible clock inputs available:

- `m_axi_mm2s_aclk` for the MM2S interface, `m_axi_s2mm_aclk` for the S2MM Interface
- `s_axi_lite_aclk` for the AXI4-Lite control interface
- `m_axi_sg_clk` for Scatter Gather Interface
- `s_axi_aclk`; common clock for the SG, MM2S and S2MM interfaces

AXI MCDMA provides two clocking modes of operation: asynchronous and synchronous. Configuring the IP for asynchronous clocks enables asynchronous mode and creates four clock domains. This allows high-performance users to run the primary data paths at a higher clock rate than the MCDMA control (for example, AXI4-Lite interface, SG Engine, MCDMA Controller) helping in FPGA placement and timing. In this mode the following clocks input pins are exposed:

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

In asynchronous mode, clocks can be run asynchronously, however `s_axi_lite_aclk` must be less than or equal to `m_axi_sg_aclk`; `m_axi_sg_aclk` must be less than or equal to the lower of `m_axi_mm2s_aclk` and `m_axi_s2mm_aclk`.

The following table shows the relationship of Input/Output ports with respect to the clocks

Table 3-1: **Relationship of Input/Output Ports to Clocks**

Clock Source	I/O Port
s_axi_lite_aclk	All s_axi_lite_* Signals mm2s_introut, mm2s_ch*_introut s2mm_introut, s2mm_ch*_introut axi_resetn
m_axi_sg_aclk	All m_axi_sg_* Signals
m_axi_mm2s_aclk	All m_axi_mm2s_* Signals All m_axis_mm2s_* Signals mm2s_prmry_reset_out_n mm2s_cntrl_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

In synchronous mode, the IP uses a single clock to run the SG, MM2S and S2MM interfaces. The IP exposes two clock ports in this case and they are `s_axi_lite_aclk` and `s_axi_aclk`.

Resets

The `axi_resetn` signal needs to be asserted a minimum of 16 of the slowest clock cycles and needs to be synchronized to `s_axi_lite_aclk`. AXI MCDMA can also be reset by writing to the 'reset' bit of control register. Resetting MM2S also resets S2MM and vice-versa.

Programming Sequence

AXI MCDMA operation requires a memory-resident data structure that holds the list of MCDMA operations to be performed. This list of instructions is organized into what is referred to as a descriptor chain. Each descriptor has a pointer to the next descriptor to be processed. The last descriptor in the chain then points back to the first descriptor in the chain. Scatter Gather operation allows a packet to be described by more than one descriptor. A typical use for this feature is to allow storing or fetching of headers from a location in memory and payload data from another location. Software applications that take advantage of this can improve throughput.

To delineate packets in a buffer descriptor chain, the Start of Frame bit (TXSOF) and End of Frame bit (TXEOF) are utilized. When the MCDMA 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 MCDMA 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 the byte count than what was specified in the descriptor, the next descriptor buffer is used to store the remainder of the receive packet. This fetching and storing process continues until the entire receive packet has been transferred. The descriptor being processed when the end of the packet is received is marked by AXI MCDMA with an RXEOF=1. This indicates to the software that the buffer associated with this descriptor contains the end of the packet. The status field of each descriptor contains the number of bytes actually transferred for that particular descriptor. The software can determine the total number of bytes transferred for the receive packet by walking from the RXSOF descriptor through the descriptor chain to the RXEOF descriptor. The Scatter Gather continues to fetch extra descriptors and store. Scatter Gather operations begin with the setting up of control registers and descriptor pointers.

The MM2S can be setup using the following programming sequence.

1. Enable the required channels. (can be also done after step 7).
2. Program the Current Descriptor (CD) registers of all the enabled channels. If the IP is configured for address_width > 32 (c_addr_width > 32), then program the corresponding MSB registers.
3. Program the CHANNEL.Fetch bit of channel control registers.

   Note: At this point the CDs cannot be re-programmed.
4. Start the MCDMA by programming MCDMA.RS bit (0x000).
5. Program the Interrupt threshold values, Enable Interrupts.
6. Program the Queue Scheduler register, if applicable.
7. Program the TD register of channels. If the IP is configured for address_width > 32 (c_addr_width > 32), then program the corresponding MSB registers. Programming the TDs of a particular channel triggers the fetching of the BDs for the respective channels.

The MCDMA schedulers start as soon as the first TD is programmed and gradually adapt as the remaining TDs are programmed.

The S2MM can be setup using the following programming sequence.

1. Enable the required channels. (can be also done after step 6).
2. Program the CD registers of the channels. If the IP is configured for address_width > 32 (c_addr_width > 32), then program the corresponding MSB registers.
3. Program the CHANNEL.Fetch bit of channel control registers.

   Note: At this point the CDs cannot be re-programmed.
4. Start the MCDMA by programming MCDMA.RS bit (0x500).
5. Program the interrupt thresholds, Enable Interrupts.
6. Program the TD register of channels. If the IP is configured for address_width > 32 (c_addr_width > 32), then program the corresponding MSB registers. Programming the TDs of a particular channel triggers the fetching of the BDs for the respective channels.

Interrupts

AXI MCDMA provides interrupt outputs per channel. Each channel has its own Status register that can be used to read the status of the channel. IOC and Delay interrupts are maintained for each channel. The Following example explains the behavior

Example:

> Let us assume that the IOC threshold of CH1 is programmed with a value of 5, IOC threshold of CH2 is programmed with 2, and IOC threshold of CH3 with 1. The MCDMA processes three packets on CH1, 2 on CH2, and 1 on CH3. This scenario will result in the IOC interrupt getting set on the interrupt outputs of CH2 and CH3. The CH1 has processed only three packets (but programmed for five) so its delay timer will start which will set the IRQ Delay Interrupt after it expiries. In this mode of operation if any channel (either S2MM or MM2S) hits an error condition, then interrupts of all the channels are asserted.

Interrupts are also generated when an error is detected. In this case, MCDMA complete all the pending transactions and stops. MCDMA has to be reset to clear the error. The IOC Interrupt is set when the MCDMA receives or transmits the number of packets as mentioned in the IOC threshold field of the Channel Control register.

The Delay Interrupt is set when there is a long gap between two packets that are received or transmitted. The delay counter starts when 'EOF' is detected (that is, TLAST on the AXI4-Stream interface). The interrupt is set If a 'SOF' (that is, a TVALID following a TLAST) is not detected before the counter expires.

Packet Drop on S2MM

AXI MCDMA has a feature to drop the packets arriving on the S2MM channel under certain circumstances. This ensures that subsequent packets are not throttled. AXI MCDMA processes and/or drops the packets as and when they arrive on the AXI4-Stream interface.

The AXI MCDMA drops an incoming packet on S2MM under the following circumstances.

- If a particular channel is not enabled and a packet arrives with that TDEST corresponding to the same channel.
- If for any channel, at the time of the packet arrival, the MCDMA runs out of BDs (that is, Current Descriptor = Tail Descriptor and the internal BD queue is empty).
- If the MCDMA is not started.

A packet that is dropped is not classified as processed.

IMPORTANT: *After the packet drop is triggered, the entire packet is dropped even if the TD is programmed to continue with the BD chain. AXI MCDMA keeps track of the packet drop for each channel and the status is provided in the Packet Drop register for the respective channel.*

Although AXI MCDMA drops packets when not started, the statistics are not updated until the MCDMA is put in run mode. A packet that is already under process is never dropped. You must ensure that enough BDs are available to service the packet to avoid throttling on AXI4-Stream interface. If the Status Stream is enabled, then for every dropped packet the corresponding Status Stream data is also dropped.

Queue Scheduler in MM2S

AXI MCDMA provides a mechanism to prioritize and send the packets on the MM2S side. It provides three different method of prioritization or scheduling of packets as follows:

Strict Priority Type of Scheduler

This type of scheduler is used when one has to transfer high-priority packets while the MCDMA is sending packets on low-priority channels. The Channel 0 (TDEST = 0) has the highest priority while Channel 15 (TDEST = 15) has the lowest priority.

The following sequence of events explain the operation of this scheduler. For Example, If MCDMA is configured for four channels (CH1, CH2, CH3, CH4), then CH1 has the highest priority while CH4 has the lowest priority. The iteration starts with CH1. The BDs of CH1 are fetched and processed, until the BD queue is completely exhausted (that is, until Current Descriptor = Tail Descriptor). While the CH1 is being serviced, other channels are idle and no BDs are processed.

When the BD queue of CH1 is exhausted, the scheduler then starts reading and processing the BDs of CH2. If CH1 BDs are not submitted, the scheduler continues to process BDs of CH2. The scheduler moves to CH3 when BD chain of CH2 is exhausted. It then moves to CH4 when BD chain of CH3 is exhausted. If at any point of time in between (while CH2, CH3 or CH4 is being serviced) BDs of CH1 are re-submitted, then the scheduler will stop processing any new BDs for the current Channel and switch back to CH1. Whenever a BD with EOF bit is detected, the MCDMA will check the status of the Channel queues. If any high priority queue is non-empty, it stops further processing of packets on the channel (that is, channel under process) and switches back to a high-priority channel. Channels that are disabled are always skipped.

As seen, re-programming of the BD chain of high-priority channels results in that channel being serviced.

Weighted Round Robin (WRR)

This type of scheduler can be used in cases where there is a need to send a predictable number of packets per channel in a round robin manner. Each channel has a weight associated with it.

The following sequence of events explains the operation of this scheduler. For example, if MCDMA is configured for three channels (CH1, CH2 and CH3), and their weights have been programmed as 3, 2 and 1 respectively. As long as BDs are available the MCDMA will process the BDs to send three packets on CH1 in succession followed by two packets on CH2 and one packet on CH3. The same sequence is repeated after processing the last packet. If any channel runs out of BDs, it is skipped until the BDs are available. Channels that are disabled are always skipped. You can re-program the weights at any time; however, the new values come into effect only when the iteration is complete. Similarly you can also enable/disable channels at any time but it will come into effect only when the iteration is over or when a BD with an EOF bit is processed.

Figure 3-2 demonstrates the WRR scheduler.

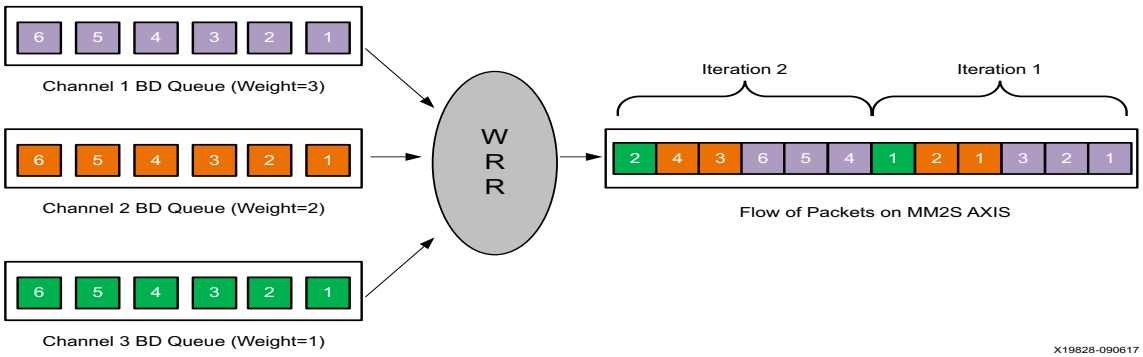

Figure 3-2: **WRR Scheduler**

Weighted Round Robin - Fair Distribution

This type of scheduler can be used in cases where there is a need to send a predictable number of packets per channel in a round robin manner but fairly distributed across the time. Each channel has a weight associate with it.

The following sequence of events explains the operation of this scheduler. For example, if MCDMA is configured for three channels (CH1, CH2 and CH3) and their weights have been programmed as 3, 2 and 1 respectively. As long as BDs are available the MCDMA will process the BDs to send one packet on CH1 in succession followed by one packet on CH2 and one packet on CH3. The weights of each channel decremented on completing each round of arbitration. In next round one packet is sent in CH1 followed by one on CH2. No packet is sent on CH3 as its weight is zero. The sequence is repeated until weights of all the channels in arbitration become zero. If any channel runs out of BDs, it is skipped and serviced again in the next iteration until the BDs are available.

Channels that are disabled are always skipped. You can re-program the weights at any time; however, the new values come into effect only when the iteration is complete. Similarly you can also enable/disable channels at any time but it will come into effect only when the iteration is over or when a BD with an EOF bit is processed.

Figure 3-3 demonstrates the WRR-FD scheduler.

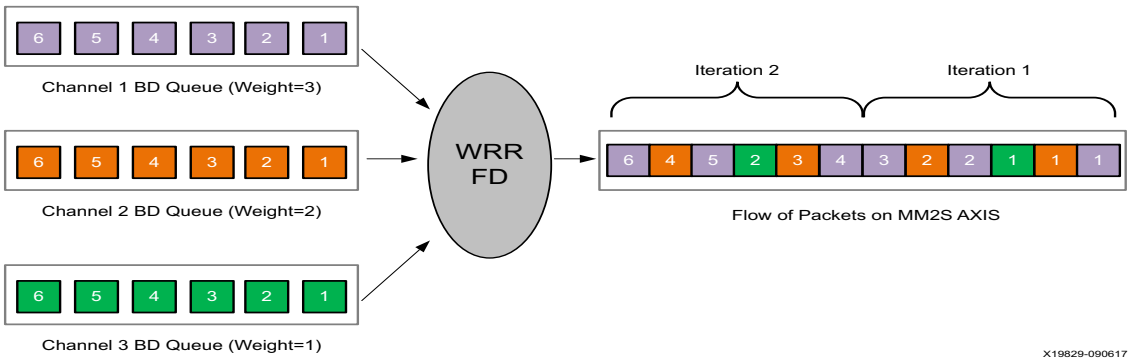

Figure 3-3: **WRR-FD Scheduler**

Note: In all the preceding cases, you can change the channel enable/disable register and weight register at any time. However, the effect of these changes is not seen until a new iteration begins and/or a BD with an EOF bit set is processed.

Note: The MM2S path implements a pipeline to process the BDs at a faster rate. Due to this, it is possible that some BDs might have been read and sent for processing. As such, the output (or change) at the streaming port will be seen only when the pipeline of BD is processed. For example, in Strict Priority mode, when the CH1 queue gets empty, BDs from CH2 are processed and pipelined. If CH1 BD chain is updated now, it will be reflected on the AXI4-Stream port only when the pipelined commands are executed.

Design Flow Steps

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

- *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 1]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 2]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 3]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 4]

Customizing and Generating the Core

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

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

To access the AXI MCDMA, do the following:

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

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 1] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref].

Note: Figures in this chapter are illustrations of the Vivado Integrated Design Environment (IDE). This layout might vary from the current version.

If you are customizing and generating the core in the Vivado IP integrator, see the *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 1] for detailed information. Vivado IDE might auto-compute certain configuration values when validating or generating the design, as noted in this section. You can view the parameter value after successful completion of the **validate_bd_design** command.

Basic Tab

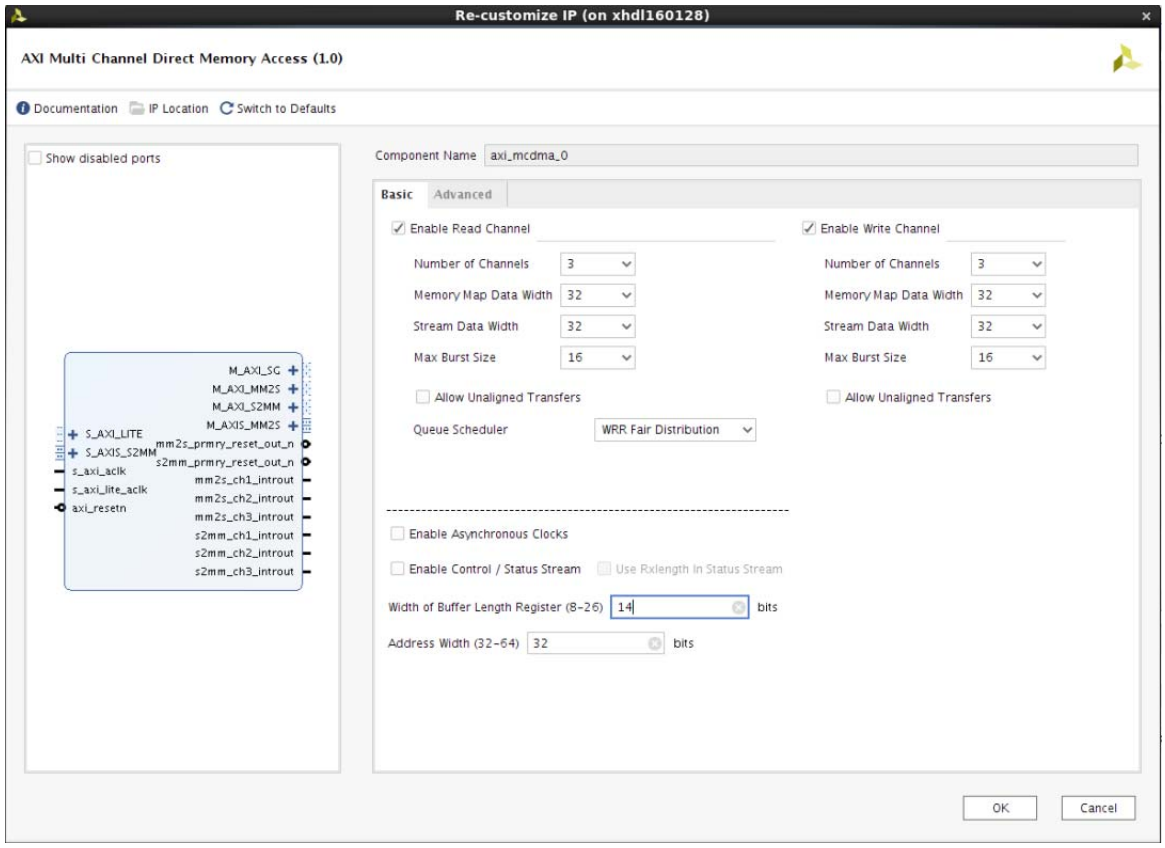

Figure 4-1: **Basic Tab**

Following are the field descriptions for the Basic tab.

Component Name

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

The following subsections describe options that affect only the MM2S Channel of the AXI MCDMA.

Enable Read 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 MCDMA core. Outputs for the MM2S channel are tied to zero and inputs are ignored by AXI MCDMA.

Number of Channels

This option specifies the number of channels from 1 to 16.

Memory Map Data Width

Data width in bits of the AXI MM2S Memory Map Read data bus. Valid values are 32, 64, 128, 256, 512 and 1,024.

Stream Data Width

Data width in bits of the AXI MM2S AXI4-Stream Data bus. This value must be equal or less than the Memory Map Data Width. Valid values are 8, 16, 32, 64, 128, 512 and 1,024.

Max Burst Size

Burst partition granularity setting. This setting specifies the maximum size of the burst cycles on the AXI4-Memory Map side of MM2S. Valid values are 2, 4, 8, 16, 32, 64, 128, and 256.

Queue Scheduler

Select the type of scheduler service required to process the MM2S Buffer Descriptors. When the number of channels is more than 1, you can select any value between Strict Priority, WRR, WRR-FD or Programmable. This value is set to Strict Priority (and cannot be altered) when the number of channels is 1.

Allow Unaligned Transfers

Enables or disables the MM2S Data Realignment Engine (DRE). When checked, the DRE is enabled and allows data realignment to the byte (8 bits) level on the MM2S Memory Map datapath. For the MM2S channel, data is read from memory. If the DRE is enabled, data reads can start from any Buffer Address byte offset, and the read data is aligned such that the first byte read is the first valid byte out on the AXI4-Stream.

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

The following options affect only the S2MM Channel of the AXI MCDMA core.

Enable Write Channel

This setting enables or disables the S2MM Channel. Enabling the S2MM Channel allows write transfers from AXI4-Stream to memory. Disabling the S2MM Channel excludes the logic from the AXI MCDMA core. Outputs for the S2MM channel are tied to zero and inputs are ignored by AXI MCDMA.

Number of Channels

This option allows you to choose a number of channels from 1 to 16.

Memory Map Data Width

Data width in bits of the AXI S2MM Memory Map Write data bus. Valid values are 32, 64, 128, 256, 512 and 1,024.

Stream Data Width

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

Max Burst Size

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

Allow Unaligned Transfers

Enables or disables the S2MM Data Realignment Engine (DRE). When checked, the DRE is enabled and allows data realignment to the byte (8 bits) level on the S2MM Memory Map datapath. For the S2MM channel, data is written to memory. If the DRE is enabled, data writes can start from any Buffer Address byte offset, and the write data is aligned such that the first valid byte received on the S2MM AXI4-Stream is written to the specified unaligned address offset.

Note: If the DRE is disabled for the respective channel, unaligned Buffer, Source, or Destination Addresses are not supported. Having an unaligned address with the DRE disabled produces undefined results. DRE support is only available for the AXI4-Stream data width setting of 64 bits and under.

Enable Asynchronous Clocks

This setting provides the ability to operate the MM2S interface `m_axi_mm2s_aclk`, S2MM interface `m_axi_s2mm_aclk` and the Scatter Gather Interface `m_axi_sg_aclk` asynchronously from each other. When asynchronous clocks are enabled, the frequency of `s_axi_lite_aclk` must be less than or equal to `m_axi_sg_aclk`. Likewise `m_axi_sg_aclk` must be less than or equal to the slower of `m_axi_mm2s_aclk` and `m_axi_s2mm_aclk`. When in synchronous mode, the IP exposes only `s_axi_lite_aclk` and `s_axi_aclk`. The `s_axi_lite_aclk` should be less than the `s_axi_aclk` frequency.

Enable Control / Status Stream

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

Use RxLength In Status Stream

If the Control/Status Stream is enabled, checking this allows AXI MCDMA to use a receive length field that is supplied by the S2MM target IP in the App4 field of the status packet. This gives AXI MCDMA a pre-determined receive byte count, allowing AXI MCDMA to command the exact number of bytes to be transferred. This option provides for a higher bandwidth solution for systems needing greater throughput. In this configuration, the S2MM target IP can supply all data bytes specified in the receive length field of status packet APP4.

Width of Buffer Length Register

This integer value specifies the number of valid bits used for the Control field buffer length and Status field bytes transferred in the Scatter/Gather descriptors. It also specifies the number of valid bits in the RX Length of the Status Stream App4 field when Use Rxlength is enabled. The length width directly correlates to the number of bytes being specified in a Scatter/Gather descriptor or the number of bytes being specified in App4.RxLength. The number of bytes is equal to 2^Length Width. So a Length Width of 26 gives a byte count of 67,108,864 bytes.

Address Width (32 - 64)

Specify the width of the Address Space. It can be any value from 32 to 64.

Advanced Tab

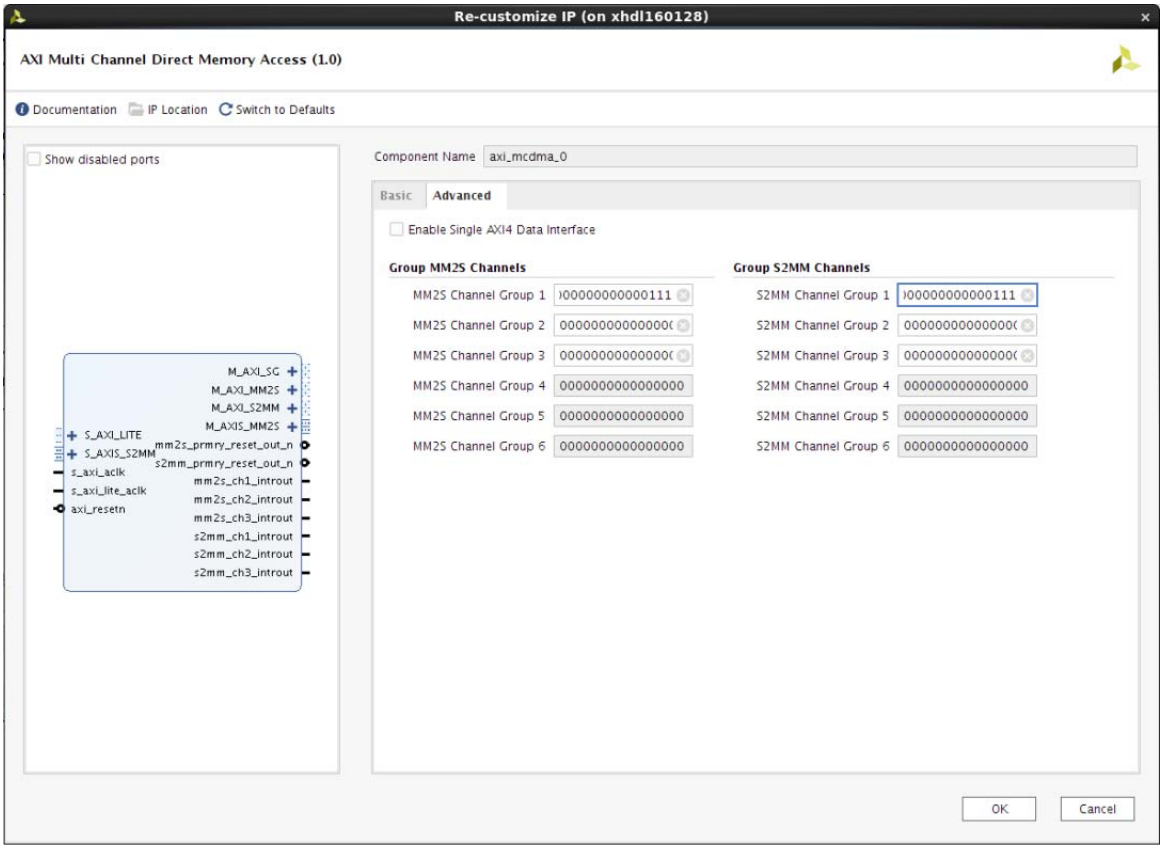

Figure 4-2: **Advanced Tab**

Following are the field descriptions for the Advanced tab.

Enable Single AXI4 Data Interface

This option combines the MM2S and S2MM AXI4 interfaces into a single interface for ease of use in the Vivado IP Integrator. The AXI4 interfaces can be combined only when they operate on the same clock and have the same data widths.

Grouping MM2S and S2MM Channels

The AXI MCDMA IP allows grouping of channels to allow an individual observer to monitor the status of particular channels. The 16 channels can be distributed in six groups. A channel can be enabled by setting the respective bit to 1. The group should be mutually exclusive, that is, one channel cannot be selected under two groups. There are six groups each for MM2S and S2MM. The channel selection is specified from LSB to MSB. For example, if six channels are enabled, specify the channel selection in bits [5:0]. In this case, the MSB bit will be ignored.

User Parameters

Table 4-1 shows the relationship between the fields in the Vivado IDE and the User Parameters (which can be viewed in the tool command language (Tcl) Console.

Table 4-1: **Vivado IDE Parameter to User Parameter Relationship**

Vivado IDE Parameter	Parameter Name	Default Value	Allowed values
Enable Asynchronous Clocks	c_prmry_is_aclk_async	FALSE	TRUE, FALSE
Enable Control/Status Stream	c_sg_include_stscntrl_strm	FALSE	TRUE, FALSE
Width of Buffer Length register	c_sg_length_width	14	8-26
Address Width (32-64)	c_addr_width	32	32-64
Number of Channels (MM2S)	c_num_mm2s_channels	1	1-16
Memory Map Data Width (MM2S)	c_m_axi_mm2s_data_width	32	32-1024
Streaming Data Width (MM2S)	c_s_axis_mm2s_data_width	32	8-1024
Max Burst Size (MM2S)	c_mm2s_burst_size	16	2-256
Scheduler Type (MM2S)	c_mm2s_scheduler	WRR	Programmable: 0 Strict Priority: 1 WRR: 2 WRR Fair Distribution: 3
Allow Unaligned Transfer (MM2S)	c_include_mm2s_dre	FALSE	TRUE, FALSE
Number of Channels (S2MM)	c_num_s2mm_channels	1	1-16
Memory Map Data Width (S2MM)	c_m_axi_s2mm_data_width	32	32-1024
Streaming Data Width (S2MM)	c_s_axis_s2mm_data_width	32	8-1024
Max Burst Size (S2MM)	c_s2mm_burst_size	16	2-256
Allow Unaligned Transfer (S2MM)	c_include_s2mm_dre	FALSE	TRUE, FALSE

Table 4-1: **Vivado IDE Parameter to User Parameter Relationship *(Cont'd)***

Vivado IDE Parameter	Parameter Name	Default Value	Allowed values
Use RxLength in Status Stream	c_sg_use_stsapp_length	FALSE	TRUE, FALSE
MM2S Channel Group 1	c_group1_mm2s	0000000000000001	16 bit binary value(1)
MM2S Channel Group 2	c_group2_mm2s	0000000000000000	16 bit binary value(1)
MM2S Channel Group 3	c_group3_mm2s	0000000000000000	16 bit binary value(1)
MM2S Channel Group 4	c_group4_mm2s	0000000000000000	16 bit binary value(1)
MM2S Channel Group 5	c_group5_mm2s	0000000000000000	16 bit binary value(1)
MM2S Channel Group 6	c_group6_mm2s	0000000000000000	16 bit binary value(1)
S2MM Channel Group 1	c_group1_s2mm	0000000000000001	16 bit binary value(2)
S2MM Channel Group 2	c_group2_s2mm	0000000000000000	16 bit binary value(2)
S2MM Channel Group 3	c_group3_s2mm	0000000000000000	16 bit binary value(2)
S2MM Channel Group 4	c_group4_s2mm	0000000000000000	16 bit binary value(2)
S2MM Channel Group 5	c_group5_s2mm	0000000000000000	16 bit binary value(2)
S2MM Channel Group 6	c_group6_s2mm	0000000000000000	16 bit binary value(2)

Notes:
1. The group selection should be mutually exclusive. Set a bit to 1 to enable that channel in a group.
2. The group selection should be mutually exclusive. Set a bit to 1 to enable that channel in a group.

Output Generation

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 1].

Constraining the Core

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

Required Constraints

This section is not applicable for this IP.

Device, Package, and Speed Grade Selections

This section is not applicable for this IP.

Clock Frequencies

This section is not applicable for this IP.

Clock Management

This section is not applicable for this IP.

Clock Placement

This section is not applicable for this IP.

Banking

This section is not applicable for this IP.

Transceiver Placement

This section is not applicable for this IP.

I/O Standard and Placement

This section is not applicable for this IP.

Simulation

For comprehensive information about Vivado simulation components, as well as information about using supported third-party tools, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 4].

IMPORTANT: *For cores targeting 7 series or Zynq®-7000 devices, UNIFAST libraries are not supported. Xilinx IP is tested and qualified with UNISIM libraries only.*

Synthesis and Implementation

For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 1].

Example Design

This chapter contains information about the example design provided in the Vivado® Design Suite. The top module instantiates all components of the core and example design that are needed to implement the design in hardware, as shown below. This includes clocking wizard, register configuration, data generator, and data checker modules.

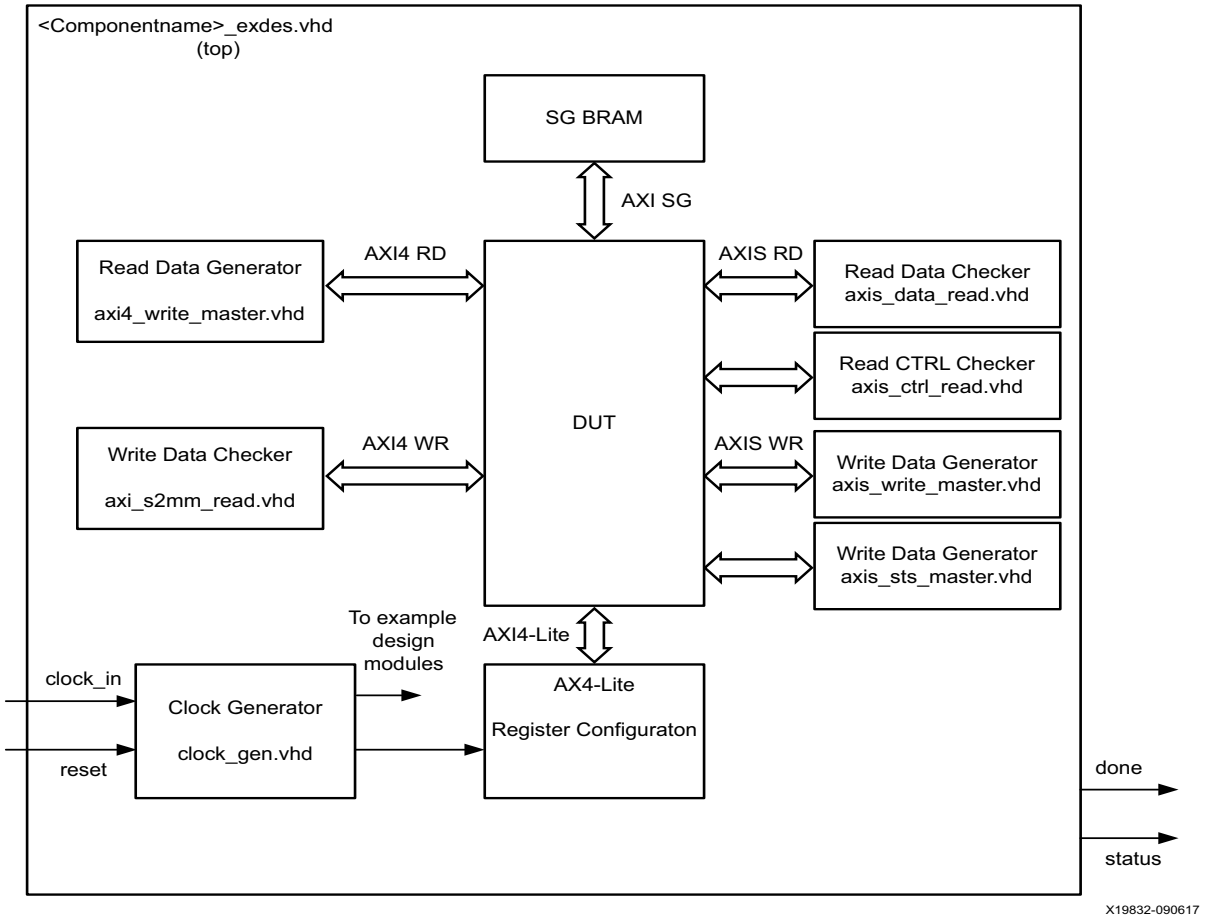

Figure 5-1: **Block Diagram**

This example design demonstrates transactions on the AXI4-Lite, AXI4, and AXI4-Stream interfaces of the device under test (DUT).

Clock Generator: MMCME2 is used to generate the clocks for the example design. This module generates various clocks needed to run the IP.

Register configuration module: This module configures the DUT registers as mentioned in Programming Sequence in Chapter 3. This module is an AXI Traffic Generator module that is configured to program the registers. For more information, refer to the *AXI Traffic Generator LogiCORE IP Product Guide* (PG125) [Ref 8].

Read path generator: This uses an AXI block RAM which is filled (with a fixed amount of transfers) after MMCME2 is locked. MM2S channel reads this AXI block RAM and transfers data to the AXI4-Stream interface.

Read path checker: This module checks the data transferred on the MM2S AXI4-Stream interface.

Read path CTRL checker: This module checks the data transferred on the MM2S AXI4-Stream Control Interface.

Write path generator: When the Write (S2MM) channel is configured, this module drives the transactions (with a fixed amount of transfers) on the S2MM AXI4-Stream interface.

Write path STS generator: This module generates the S2MM STS data stream.

Write path checker: This module checks the data received on the AXI4 interface. Data received on the AXI4 interface is also written into another AXI block RAM.

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

Implementing the Example Design

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

After following the steps described in Chapter 4, Design Flow Steps, implement the example design as follows:

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

3. In the Flow Navigator (left-side pane), click Run Implementation and follow the directions. In the current project directory, a new project with the name _ex0 is created and the files are delivered in that directory. This directory and its subdirectories contain all the source files that are required to create the AXI MCDMA controller example design.

Simulating the Example Design

Using the AXI MCDMA example design (delivered as part of the AXI MCDMA), the behavior of the AXI MCDMA can be quickly simulated and observed. The simulation script compiles the AXI MCDMA example design and supporting simulation files. It then runs the simulation and checks to ensure that it completed successfully.

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

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

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

Test Bench for the Example Design

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

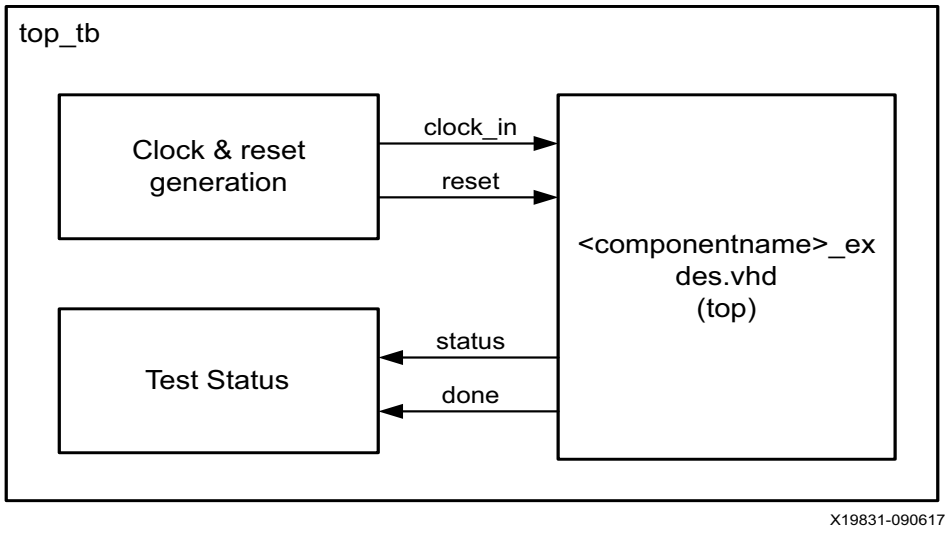

Figure 5-2: **Test Bench**

Figure 5-2 shows the test bench for the AXI MCDMA example design. The top-level test bench generates a 200 MHz differential clock and drives an initial reset to the example design.

Upgrading

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

Debugging

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

Finding Help on Xilinx.com

To help in the design and debug process when using the AXI MCDMA core, the Xilinx Support web page contains key resources such as product documentation, release notes, answer records, information about known issues, and links for obtaining further product support.

Documentation

This product guide is the main document associated with the AXI MCDMA core. This guide, along with documentation related to all products that aid in the design process, can be found on the Xilinx Support web page or by using the Xilinx Documentation Navigator.

Download the Xilinx Documentation Navigator from the Downloads page. For more information about this tool and the features available, open the online help after installation.

Answer Records

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

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

- Product name
- Tool message(s)
- Summary of the issue encountered

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

Master Answer Records for the AXI MCDMA core

AR: 69734

Technical Support

Xilinx provides technical support in the Xilinx Support web page for this LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support if you do any of the following:

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

To contact Xilinx Technical Support, navigate to the Xilinx Support web page.

Vivado Design Suite Debug Feature

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

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

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

Several internal signal are marked as debug signals. These can be easily added to the logic analyzer.

See the *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 6].

Hardware Debug

Hardware issues can range from link bring-up to problems seen after hours of testing. This section provides debug steps for common issues. The Vivado Design Suite debug feature is a valuable resource to use in hardware debug. Some of the common problems encountered and possible solutions follow.

- You have programmed your BD ring but nothing seems to work. The register programming sequence has to be followed to start the MCDMA. See Programming Sequence in Chapter 3 and Descriptor Management in Chapter 2.
- Internal Error/Error bits set in the Status register
  - An internal error will be set when the bytes to transfer (BTT) specified in the descriptor is 0.
  - SG internal error will be set if the fetched BD is a completed BD.
  - Other error bits like Decode Error or Slave Error will also be set based on the response from Interconnect or Slave.
- You are reading data from a location, but the data does not seem to be in order. Verify if the start address location is aligned or unaligned. If it is not aligned, ensure that the DRE is enabled while configuring MCDMA.

Also 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.

Appendix C

Additional Resources and Legal Notices

Xilinx Resources

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

Documentation Navigator and Design Hubs

Xilinx Documentation Navigator provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open the Xilinx Documentation Navigator (DocNav):

- From the Vivado® IDE, select **Help > Documentation and Tutorials**.
- On Windows, select **Start > All Programs > Xilinx Design Tools > DocNav**.
- At the Linux command prompt, enter `docnav`.

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

- In the Xilinx Documentation Navigator, click the **Design Hubs View** tab.
- On the Xilinx website, see the Design Hubs page.

Note: For more information on Documentation Navigator, see the Documentation Navigator page on the Xilinx website.

References

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

1. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
2. *Vivado Design Suite User Guide: Designing with IP* (UG896)
3. *Vivado Design Suite User Guide: Getting Started* (UG910)
4. *Vivado Design Suite User Guide - Logic Simulation* (UG900)
5. *ISE to Vivado Design Suite Migration Guide* (UG911)
6. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
7. *Vivado Design Suite User Guide - Implementation* (UG904)
8. *AXI Traffic Generator LogiCORE IP Product Guide* (PG125)
9. *AXI Reference Guide* (UG1037)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
10/04/2017	1.0	Initial public release.

Please Read: Important Legal Notices

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

AUTOMOTIVE APPLICATIONS DISCLAIMER

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