LogiCORE IP AXI Video Direct Memory Access v5.02.a

Product Guide

PG020 July 25, 2012

XILINX®

XILINX®

Table of Contents

SECTION I: SUMMARY

Introduction

The Advanced eXtensible Interface Video Direct Memory Access (AXI VDMA) core is a soft Xilinx Intellectual Property (IP) core providing high-bandwidth direct memory access between memory and AXI4-Stream video type target peripherals including peripherals which support AXI4-Stream Video Protocol as described in the Video IP: AXI Feature Adoption section of the *AXI Reference Guide* (UG761). Initialization, status, and management registers are accessed through an AXI4-Lite slave interface.

Features

- AXI4 Compliant
- Primary AXI4 Memory Map data width support of 32, 64, 128, 256, 512, and 1024 bits
- Primary AXI4-Stream data width support of multiples of 8 up to 1024 bits
- Register Direct Mode
- Optional independent Scatter Gather Direct Memory Access (DMA) support
- Optional Data Re-Alignment Engine
- Optional Genlock Synchronization
- Optional Line Buffers and Store-And-Forward
- Independent, asynchronous channel operation
- Dynamic clock frequency change of AXI4-Stream interface clocks
- Dynamic line buffer threshold
- Optional flush on frame sync
- Optional frame advancement on error
- Optional fsync crossbar, 32 fstores, and internal Genlock

LogiCORE™ IP Facts Table	
Core Specifics	
Supported Device Family(1)	Zynq™-7000(2), Virtex®-7, Kintex™-7, Artix™-7, Virtex-6, Spartan®-6
Supported User Interfaces	AXI4, AXI4-Lite, AXI4-Stream
Resources	See Table 2-4 and Table 2-5.
Provided with Core	
Design Files(3)	ISE®: VHDL Vivado™: VHDL
Example Design	XAPP739, XAPP740, XAPP741, XAPP742
Test Bench	Not Provided
Constraints File	Not Provided
Simulation Model	Not Provided
Supported S/W Drivers(4)	Standalone and Linux
Tested Design Flows(5)	
Design Entry	Embedded Development Kit (EDK) 14.2 ISE Design Suite 14.2 Vivado™ Design Suite 2012.2(6)
Simulation	ModelSim
Synthesis	Xilinx Synthesis Technology (XST) Vivado Synthesis
Support	
Provided by Xilinx @ www.xilinx.com/support	

1. For a complete list of supported EDK derivative devices, see Embedded Edition Derivative Device Support.
2. Supported in ISE Design Suite implementations only.
3. Contains few Verilog files. Top level is VHDL.
4. Standalone driver information can be found in the EDK or SDK installation directory.

   See xilinx_drivers.htm in <install_directory>/doc/usenglish. Linux OS and driver support information is available from wiki.xilinx.com.
5. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.
6. Supports only 7 series devices.

Overview

Many video applications need a frame buffer to handle things like rate changes or changes to the image dimensions (such as, scaling). The AXI VDMA is designed to allow for efficient high-bandwidth access between AXI4-Stream video data and AXI4 Memory Mapped data, which is typically connected to external storage such as an external DDR2 memory. This includes peripherals supporting the AXI4-Stream Video Protocol as described in the Video IP: AXI Feature Adoption section of the *AXI Reference Guide* (UG761).

The AXI VDMA core has four AXI4 interfaces:

- AXI4-Lite Slave
- AXI4 Read Master
- AXI4 Write Master
- AXI4 Scatter Gather Read Only Master.

Associated with the memory map interfaces are two AXI4-Stream interfaces: AXI Memory Map to Stream (MM2S) Stream Master, AXI4-Stream to Memory Map (S2MM) Stream Slave. Optional Genlock and Video Frame Sync interfaces are also provided for each channel. Register access and configuration are provided through the AXI4-Lite slave interface. The register module provides control and status for DMA operations.

Primary high-speed DMA data movement between system memory and the stream target is through the AXI4 Read Master to AXI MM2S Stream Master and AXI S2MM Stream Slave to AXI4 Write Master. The AXI DataMover is used for high throughput transfer of data from memory to stream and from stream to memory. The MM2S channel and S2MM channel operate independently and in a full duplex like method. The AXI DataMover provides the AXI VDMA with a 4 KB address boundary protection and automatic burst partitioning. It also provides the ability to queue multiple transfer requests using nearly the full bandwidth capabilities of the AXI4-Stream buses. Furthermore, the AXI DataMover provides byte-level data realignment, allowing memory reads and writes to any byte offset location.

Register Direct Mode

The AXI VDMA provides a Register Direct Mode that allows the processor to directly control the operation of the core. In this mode the video parameter registers and start address registers are accessible through the Slave AXI4-Lite control interface. Figure 1-1 and Figure 1-2 illustrate the AXI VDMA configured for Register Direct Mode.

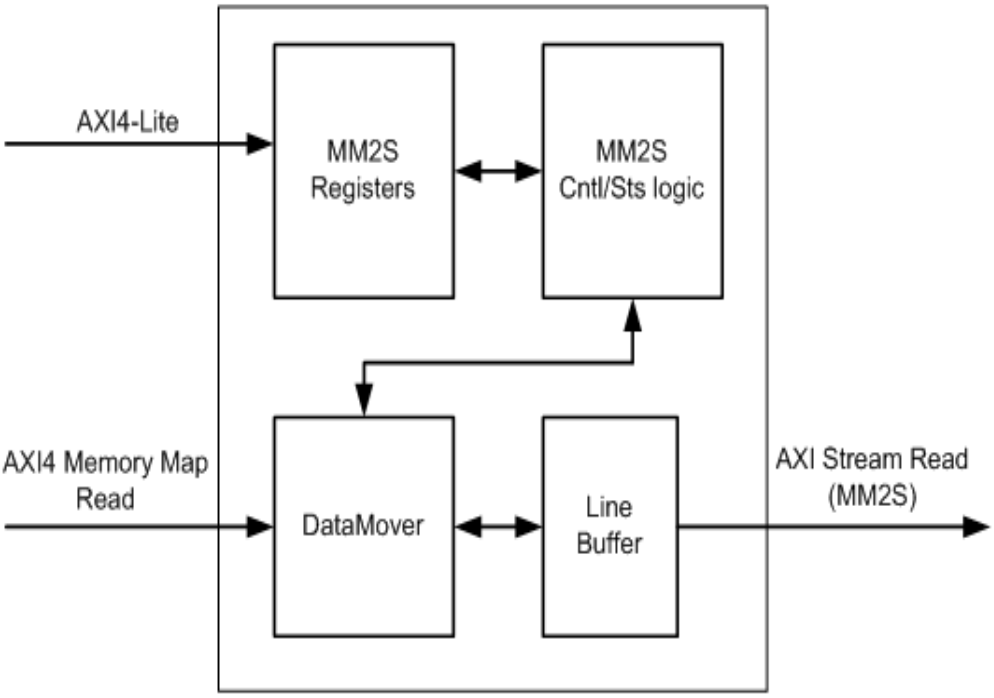


Figure 1-1: **AXI4 Memory Map to AXI4-Stream Read**

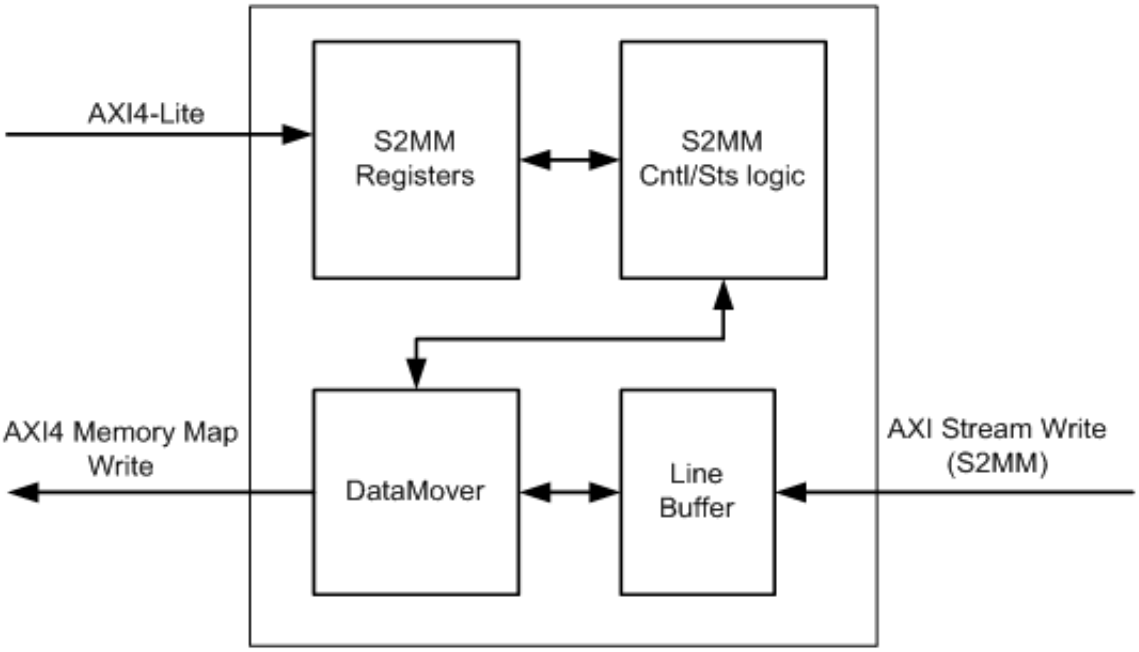


Figure 1-2: **AXI4-Stream to AXI4 Memory Map Write**

Scatter Gather Mode

The AXI VDMA provides an optional Scatter Gather Mode for off-loading processor management tasks to hardware. The Scatter Gather Engine fetches and updates buffer descriptors from system memory through the AXI4 Memory Map Scatter Gather Read/Write Master interface.

Feature Summary

AXI4 Compliant

The AXI VDMA core is fully compliant with the AXI4 Memory Map interface, AXI4-Stream interface and AXI4-Lite interface. The AXI4-Stream also supports the Video Protocol as described in the "Video IP: AXI Feature Adoption" section of the *AXI Reference Guide* (UG761).

AXI4 Memory Map Data Width

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

AXI4-Stream Data Width

The AXI VDMA core supports the primary AXI4-Stream data bus width of multiples of 8 bits up to 1024 bits. The AXI4-Stream data width must be less than or equal to the AXI4 Memory Map data width for the respective channel.

Register Direct Mode

The AXI VDMA core supports register direct mode in which the transfer descriptors are placed in the control register map along with the video-specific registers. In this mode, the independent Scatter Gather AXI4-Memory Map bus is not used for fetching and updating of transfer descriptors.

Scatter Gather Mode

The AXI VDMA core supports fetching and updating of transfer descriptors through the independent Scatter Gather AXI4-Memory Map bus. This allows descriptor placement to be in any memory-mapped location separate from data buffers.

Data Realignment Engine

The AXI VDMA core supports the optional Data Realignment Engine (DRE). When the DRE is enabled, the DRE Width matches the associated Payload Stream interface width up to 64 bits.

Genlock Synchronization

The AXI VDMA core supports Genlock synchronization. Each channel of AXI VDMA can be designed to operate as either a Genlock Master/Slave or Dynamic Genlock Master/Slave. By using this feature, the master and slave are kept in sync by not allowing both to use the same buffer at the same time.

The AXI VDMA core also supports an optional internal Genlock Bus. This allows an internal connection of the Genlock bus, which provides the option to not connect a Genlock bus externally between mm2s and s2mm channels. A DMACR register control bit (bit 7) is also added to allow dynamic selection of internal or external Genlock for channels configured as a Genlock Slave.

Line Buffers and Store and Forward

The AXI VDMA core supports an optional line buffer that can be utilized to prevent memory controller throttling from causing inner packet throttling on the stream interface. Line buffer parameters like empty and full signals are driven out of the AXI VDMA core for Video IP use.

The AXI VDMA core also supports the optional Store-And-Forward feature. On MM2S, this prevents the channel from requesting more read data than can be held in the Store-And-Forward buffer. On S2MM this prevents the channel from issuing write requests when there is not enough data in the Store-And-Forward buffer to complete the write.

Asynchronous Channels

The AXI VDMA core supports asynchronous clock domains for AXI4-Lite, AXI Scatter Gather (SG), S2MM AXI4-Stream interface, MM2S AXI4-Stream interface, S2MM AXI4 Memory Map interface and MM2S AXI4 Memory Map interface.

Frame Sync on TUSER0

The AXI VDMA supports an optional TUSER bus on both MM2S and S2MM AXIS interfaces with TUSER(0) being used for a Start of Frame (SOF) or external frame sync. When enabled (C_MM2S_SOF_ENABLE=1), MM2S channel will drive frame sync out on `m_axis_mm2s_tuser(0)`. When enabled (C_S2MM_SOF_ENABLE=1), S2MM channel will sync to frame sync in on `s_axis_s2mm_tuser(0)`. For more information, see the Video IP: AXI Feature Adoption section of the UG761 *AXI Reference Guide*.

Frame Sync Crossbar

This feature allows routing of an AXI VDMA frame sync source to both channels. Control bits are added to the DMACR (bits 5 and 6) of both channels for selecting the respective channels frame sync source. This feature is only available when the channel uses external frame sync.

32 Frame Stores

Support for the number of frame stores has been increased from 16 to 32 for each channel. For SG =1 mode, it increases the maximum length of the descriptor chain from 16 to 32 (for each channel). For SG=0 mode, it increases the maximum value of Frame Store Start Address registers from 16 to 32 (for each channel). In this mode, MM2S_REG_INDEX and S2MM_REG_INDEX are added to create another set of register bank of 16 frame stores. This is done to keep it backward compatible with AXI VDMA previous versions.

Dynamic Clock Frequency Change of AXI4-Stream Interface Clocks

The AXI VDMA core allows you to change the primary datapath clocks dynamically to support different video resolutions without rebuilding the system.

Dynamic Line Buffer Threshold

This feature allows the almost empty and almost full threshold values to be dynamically changed by accessing new threshold registers.

Flush on Frame Sync

The flush on frame sync feature allows AXI VDMA to reset internal states and flush transfer data on frame sync for certain error conditions. This allows AXI VDMA to restart transfers at the beginning of the next new frame after DMA Internal error detection instead of halting the channel. This feature is added for both MM2S and S2MM channels independently.

Optional Frame Advancement on Error

When an error is detected in a particular frame, this optional feature allows the user to let the frame number advance on the next frame sync or not advance and reuse the errored frame's frame number.

Applications

The AXI VDMA core provides high-speed data movement between system memory and AXI4-Stream Video Protocol Video IP.

Unsupported Features

The following AXI4 features are not supported by the AXI VDMA design.

- User signals on AXI4 Memory Map Interface
- Locked transfers
- Exclusive transfers
- FIXED and WRAP Burst transfers

Licensing and Ordering Information

This Xilinx LogiCORE™ IP module is provided at no additional cost with the Xilinx Vivado™ Design Suite and ISE® Design Suite tools 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 about pricing and availability of other Xilinx LogiCORE IP modules and tools, contact your local Xilinx sales representative.

Product Specification

Performance

This section provides information about the performance of the AXI VDMA. Streaming side of AXI VDMA is looped back using shim logic. The block diagram shown in Figure 2-1 shows the configuration of the system that is used to report the frequency numbers in Table 2-1.

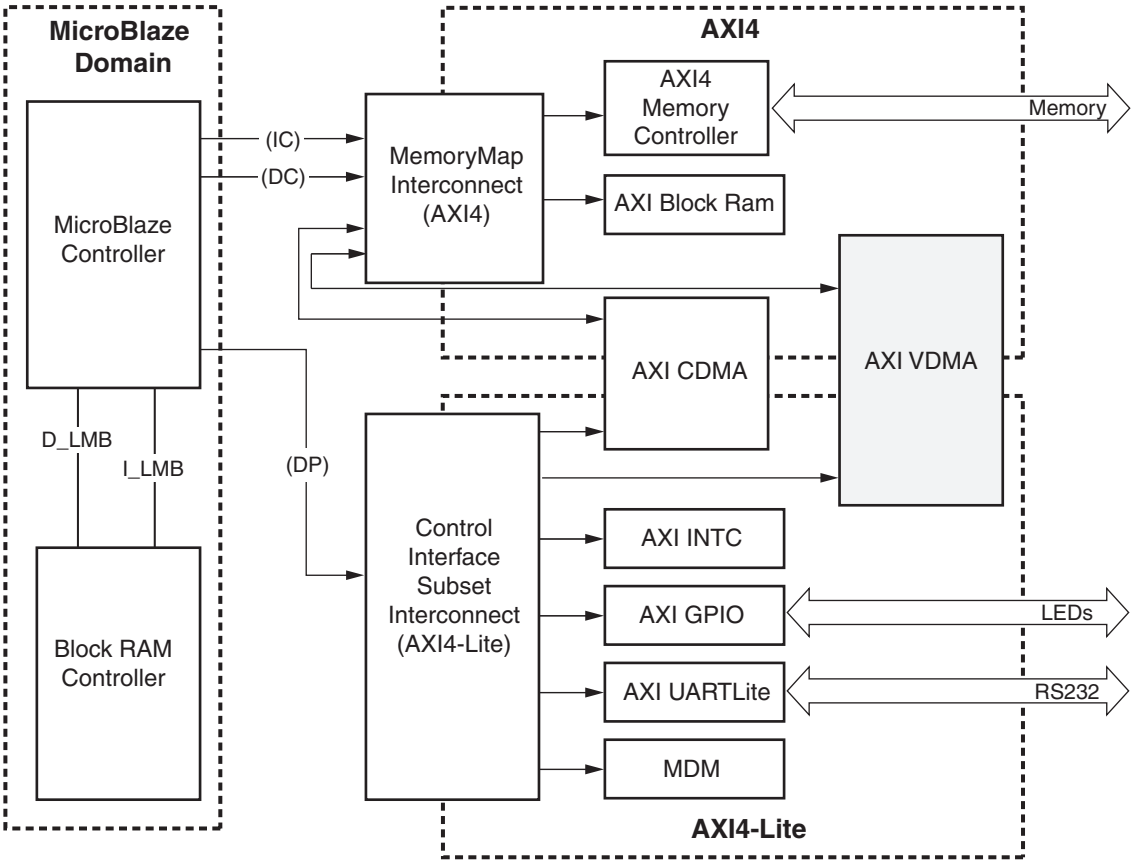


Figure 2-1: **FPGA System Configuration Used for Generating System Performance Information**

Maximum Frequencies

The target Field Programmable Gate Array (FPGA) was filled with logic to drive the Lookup Table (LUT) and block Random Access Memory (RAM) utilization to approximately 70% and the I/O utilization to approximately 80%. Using the default tool options and the slowest speed grade for the target FPGA, the resulting target FMAX numbers are shown in Table 2-1.

Table 2-1: **Maximum Frequencies**

Family	Device	Speed Grade	Fmax[1]		
			AXI4	AXI4-Lite	AXI4-Stream
Spartan-6[2]	xc6slx45t	-2	150 MHz	100 MHz	150 MHz
Virtex-6[3]	xc6vlx240t	-1	200 MHz	150 MHz	200 MHz
Virtex-7[3]	xc7vx485tffg1761	-1	200 MHz	150 MHz	200 MHz
Kintex-7[3]	xc7k325tffg900	-1	200 MHz	150 MHz	200 MHz
Zynq-7000[3]	xc7z030fbg676	-1	200 MHz	150 MHz	200 MHz

Notes:

1. Fmax numbers represent both MM2S and S2MM channel clocks.
2. MicroBlaze™ processor frequency is 80 MHz.
3. MicroBlaze processor frequency is 150 MHz.
4. For better performance have AXI4 clock equal to greater than AXI4-Stream clock

Latency and Throughput

Table 2-2 and Table 2-3 describe the throughput and latency for the AXI VDMA. The tables provide performance information for a typical configuration. The throughput test consisted of eight video frames for each channel with each descriptor describing a 1000 lines at 1000 bytes per line per frame (~1 MB) and each channel operating simultaneously (full duplex). Throughput is measured from completion of descriptor fetches (DMACR.Idle = 1) to frame count interrupt assertion. Latency is measured on both the mm2s and s2mm path. Table 2-3 shows the AXI VDMA core latency cycles only and does not include system dependent latency or throttling.

- AXI VDMA Configuration
  - C_USE_FSYNC = 0
  - C_NUM_FSTORES = 8
  - C_M_AXI_MM2S_DATA_WIDTH = 32 and C_M_AXI_S2MM_DATA_WIDTH = 32
  - C_M_AXIS_MM2S_TDATA_WIDTH = 32 and C_S_AXIS_S2MM_TDATA_WIDTH = 32
  - C_MM2S_MAX_BURST_LENGTH = 16 and C_S2MM_MAX_BURST_LENGTH = 16
  - C_MM2S_GENLOCK_MODE = 0 and C_S2MM_GENLOCK_MODE = 0
  - C_MM2S_LINEBUFFER_DEPTH = 0 and C_S2MM_LINEBUFFER_DEPTH = 0

Table 2-2: **AXI VDMA Throughput (Synchronous Mode)**

Channel	Clock Frequency (in MHz)	Frame Size (In Bytes)	Maximum Total Data Throughput (MBytes/sec)	Percent of Theoretical
MM2S	80	1 MB	287	89.68
	150	1 MB	539	89.83
S2MM	80	1 MB	290	90.6
	150	1 MB	542	90.3

Table 2-3: **AXI VDMA Latency (Free Run Mode)**

Description	Clocks
MM2S Channel	
mm2s_fsync_out to m_axi_mm2s_arvalid	14
m_axi_mm2s_rvalid to m_axis_mm2s_tvalid	4
last m_axis_mm2s_tlast to next mm2s_fsync_out	8
S2MM Channel	
s_axis_s2mm_tvalid to m_axi_s2mm_awvalid	14
m_axi_s2mm_awvalild and m_axi_s2mm_awready=1 to m_axi_s2mm_wvalid	2
last m_axi_s2mm_wlast to next s2mm_fsync_out	11

HSync Period

- HSync period requirement for 1920x1080p frame = 14.81 us.
- HSync period achieved by AXI VDMA in 32-bit streaming data width (SOF - TLAST) = 9.59 us
- HSync period achieved by AXI VDMA in 24-bit streaming data width (SOF - TLAST) = 12.79 us

Resource Utilization

Resources required for the AXI VDMA core have been estimated for Virtex®-7, Kintex™-7, Virtex-6, and Spartan®-6 devices. These values were generated using the Xilinx 14.2 EDK tools. They are derived from post-synthesis reports and can change during MAP and PAR. Table 2-4 show 33 cases that are used for resource estimation.

Table 2-5 shows resource estimates for Virtex-7, Kintex-7, Virtex-6, and Spartan-6 devices for the 33 cases in Table 2-4.

Note: Resource requirements for Artix™-7 and Zynq™-7000 devices are similar to Kintex-7 or Virtex-7 FPGAs as all 7 series devices are based on the same architecture.

Table 2-4: **Resource Estimations for 33 Cases**

	C_INCLUDE_MM2S	C_INCLUDE_S2MM	C_M_AXI_MM2S_DATA_WIDTH	C_M_AXI_S2MM_DATA_WIDTH	C_M_AXIS_MM2S_TDATA_WIDTH	C_S_AXIS_S2MM_TDATA_WIDTH	C_MM2S_MAX_BURST_LENGTH	C_S2MM_MAX_BURST_LENGTH	C_INCLUDE_MM2S_DRE	C_INCLUDE_S2MM_DRE	C_MM2S_LINEBUFFER_DEPTH	C_S2MM_LINEBUFFER_DEPTH	C_USE_FSYNC	C_NUM_FSTORES	C_MM2S_GENLOCK_MODE	C_S2MM_GENLOCK_MODE	C_INCLUDE_SG	C_ENABLE_VIDPRMTR_READS	C_PRMRY_IS_ACLK_ASYNC	C_INCLUDE_MM2S_SF	C_INCLUDE_S2MM_SF	C_FLUSH_ON_FSYNC	C_S2MM_SOF_ENABLE	C_MM2S_SOF_ENABLE
case1	1	1	32	32	8	8	16	16	1	1	0	0	1	3	1	1	0	1	0	1	1	0	1	1
case2	0	1	-	64	-	16	-	16	-	1	-	0	1	3	-	0	0	1	0	-	1	0	-	1
case3	0	1	-	64	-	16	-	256	-	1	-	0	1	3	-	0	0	1	0	-	1	0	-	1
case4	1	0	64	-	16	-	16	-	1	1	0	-	1	3	0	-	0	1	0	1	-	0	1	-
case5	1	0	64	-	16	-	256	-	1	1	0	-	1	3	0	-	0	1	0	1	-	0	1	-
case6	1	1	256	256	16	16	256	256	1	1	0	0	1	3	1	1	0	1	0	1	1	0	1	1
case7	0	1	-	128	-	24	-	16	1	1	-	0	1	3	-	0	0	1	0	-	1	0	-	1
case8	0		-	128	-	24	-	256	1	1	-	1	1	3	-	0	0	1	0	-	1	0	-	1
case9	1	0	128	-	24	-	16	-	1	1	0	-	1	3	0	-	0	1	0	1	-	0	1	-
case10	1	0	128	-	24	-	256	-	1	1	0	-	1	3	0	-	0	1	0	1	-	0	1	-
case11	1	1	256	256	24	24	16	16	1	1	0	0	1	3	1	1	0	1	0	1	1	0	1	1
case12	1	1	256	256	24	24	256	256	1	1	0	0	1	3	1	1	0	1	0	1	1	0	1	1
case13	1	1	1024	1024	64	64	256	256	1	1	0	0	1	3	1	1	0	1	0	1	1	0	1	1
case14	0	1	-	64	-	16	-	16	1	1	-	2048	1	3	-	0	0	1	0	-	1	1	-	1
case15	0	1	-	64	-	16	-	256	1	1	-	2048	1	3	-	0	0	1	0	-	1	1	-	1
case16	1	0	64	-	16	-	16	-	1	1	2048	-	1	3	0	-	0	1	0	1	-	1	1	-
case17	1	0	64	-	16	-	256	-	1	1	2048	-	1	3	0	-	0	1	0	1	-	1	1	-
case18	1	1	256	256	16	16	256	256	1	1	2048	2048	1	3	1	1	0	1	0	1	1	1	1	1

Table 2-4: **Resource Estimations for 33 Cases *(Cont'd)***

	C_INCLUDE_MM2S	C_INCLUDE_S2MM	C_M_AXI_MM2S_DATA_WIDTH	C_M_AXI_S2MM_DATA_WIDTH	C_M_AXIS_MM2S_TDATA_WIDTH	C_S_AXIS_S2MM_TDATA_WIDTH	C_MM2S_MAX_BURST_LENGTH	C_S2MM_MAX_BURST_LENGTH	C_INCLUDE_MM2S_DRE	C_INCLUDE_S2MM_DRE	C_MM2S_LINEBUFFER_DEPTH	C_S2MM_LINEBUFFER_DEPTH	C_USE_FSYNC	C_NUM_FSTORES	C_MM2S_GENLOCK_MODE	C_S2MM_GENLOCK_MODE	C_INCLUDE_SG	C_ENABLE_VIDPRMTR_READS	C_PRMRY_IS_ACLK_ASYNC	C_INCLUDE_MM2S_SF	C_INCLUDE_S2MM_SF	C_FLUSH_ON_FSYNC	C_S2MM_SOF_ENABLE	C_MM2S_SOF_ENABLE
case19	0	1	-	128	-	32	-	16	1	1	-	2048	1	3	-	0	0	1	0	-	1	1	-	1
case20	0	1	-	128	-	32	-	256	1	1	-	2048	1	3	-	0	0	1	0	-	1	1	-	1
case21	1	0	128	-	32	-	16	-	1	1	2048	2048	1	3	0	-	0	1	0	1	-	1	1	-
case22	1	0	128	-	32	-	256	-	1	1	2048	-	1	3	0	-	0	1	0	1	-	1	1	-
case23	1	1	256	256	32	32	16	16	1	1	2048	2048	1	3	1	1	0	1	0	1	1	1	1	1
case24	1	1	256	256	32	32	256	256	1	1	2048	2048	1	3	1	1	0	1	1	1	1	1	1	1
case25	1	1	256	256	48	48	32	32	1	1	0	0	1	3	1	1	0	1	1	1	1	1	1	1
case26	1	1	256	256	48	48	256	256	1	1	0	0	1	3	1	1	0	1	1	1	1	1	1	1
case27	0	1	-	256	-	48	-	16	1	1	-	0	1	3	-	1	0	0	0	-	1	1	-	0
case28	0	1	-	256	-	48	-	256	1	1	-	2048	1	3	-	1	0	1	0	-	1	1	-	0
case29	1	0	256	-	48	-	16	-	1	1	0	-	1	3	1	-	0	0	0	1	-	1	0	-
case30	1	0	256	-	48	-	256	-	1	1	2048	-	1	3	1	-	0	1	0	1	-	1	0	-
case31	1	1	64	64	64	64	32	32	1	1	0	0	1	3	1	0	1	1	0	1	1	1	0	0
case32	1	1	64	64	64	64	32	32	1	1	0	0	1	3	1	0	1	1	1	1	1	1	0	0
case33	1	1	1024	1024	1024	1024	256	256	0	0	65536	65536	1	3	1	0	1	1	1	1	1	1	0	0

Table 2-5: **Resource Estimates for Virtex-7, Kintex-7, Virtex-6, and Spartan-6 Devices**

	Kintex-7				Virtex-7				Spartan-6				Virtex-6			
	Number of Occupied Slices	Number of Slice Registers	Number of Slice LUTs	Number of Block RAMs	Number of occupied Slices	Number of Slice Registers	Number of Slice LUTs	Number of Block RAMs	Number of occupied Slices	Number of Slice Registers	Number of Slice LUTs	Number of Block RAMs	Number of occupied Slices	Number of Slice Registers	Number of Slice LUTs	Number of Block RAMs
case1	1162	3139	2542	4	1223	3139	2498	4	1090	3145	2366	5	1216	3136	2524	4
case2	666	1993	1594	2	753	1993	1504	2	664	1993	1494	3	773	1992	1459	2
case3	680	2037	1624	5	760	2037	1539	5	696	2038	1533	9	757	2036	1507	5
case4	671	1650	1202	3	686	1650	1198	3	539	1647	1247	4	644	1648	1207	3
case5	663	1689	1312	6	621	1689	1354	6	572	1692	1265	10	645	1687	1246	6
case6	1551	4547	3646	19	1389	4547	3779	19	1578	4570	3342	34	1671	4543	3621	19
case7	977	2809	1925	3	990	2809	1905	3	844	2812	1882	6	941	2808	1953	3
case8	908	2855	2065	9	886	2855	2100	9	885	2864	1957	17	1001	2854	1977	9
case9	810	2219	1595	4	826	2219	1563	4	734	2218	1464	7	828	2217	1510	4
case10	866	2260	1562	10	759	2260	1710	10	748	2283	1559	19	849	2258	1554	10
case11	1859	5528	4040	11	1840	5528	4152	11	1783	5537	3681	21	1912	5526	3896	11
case12	1919	5594	4082	19	2026	5594	3953	19	1815	5631	3788	36	1826	5592	4080	19
case13	2811	9071	7059	36	2915	9071	7012	36	2581	9131	6880	68	2897	9070	6923	36
case14	869	2264	1671	3	852	2264	1711	3	811	2265	1665	5	902	2263	1699	3
case15	858	2300	1756	6	812	2300	1791	6	799	2303	1721	11	839	2299	1783	6
case16	676	1678	1281	3	737	1678	1248	3	605	1674	1214	5	675	1676	1260	3
case17	663	1717	1348	6	739	1717	1291	6	589	1719	1372	11	622	1714	1314	6
case18	1845	5170	3884	20	2032	5170	3754	20	1782	5212	3942	38	1916	5169	3908	20
case19	997	2804	2010	4	932	2804	2053	4	887	2812	2046	7	976	2803	2028	4
case20	975	2841	2127	10	1011	2841	2083	10	953	2865	2171	19	1092	2840	2115	10
case21	719	1860	1450	4	737	1860	1459	4	623	1858	1377	7	738	1858	1445	4
case22	774	1900	1474	10	749	1900	1485	10	657	1923	1472	19	760	1898	1448	10
case23	1848	5385	4082	12	1955	5385	3960	12	1845	5394	3779	22	1881	5384	4047	12
case24	2467	6810	4390	20	2390	6810	4480	20	2277	6846	4444	38	2424	6809	4489	20
case25	2849	8746	5389	14	2951	8746	5280	14	2632	8754	5598	24	2907	8745	5290	14
case26	2838	8786	5598	22	2976	8786	5403	22	2641	8822	5806	40	2846	8784	5645	22
case27	1375	4608	3008	7	1450	4608	2806	7	1282	4612	2968	12	1425	4606	2924	7
case28	1361	4650	3158	11	1529	4650	2999	11	1468	4668	2959	20	1516	4648	3020	11
case29	910	2946	2260	7	1053	2946	2038	7	933	2948	1956	12	985	2942	2148	7
case30	1123	2988	2236	11	1124	2988	2228	11	963	3009	2163	20	1081	2986	2144	11

Table 2-5: **Resource Estimates for Virtex-7, Kintex-7, Virtex-6, and Spartan-6 Devices *(Cont'd)***

	Kintex-7				Virtex-7				Spartan-6				Virtex-6			
	Number of Occupied Slices	Number of Slice Registers	Number of Slice LUTs	Number of Block RAMs	Number of occupied Slices	Number of Slice Registers	Number of Slice LUTs	Number of Block RAMs	Number of occupied Slices	Number of Slice Registers	Number of Slice LUTs	Number of Block RAMs	Number of occupied Slices	Number of Slice Registers	Number of Slice LUTs	Number of Block RAMs
case31	2478	7147	4493	8	2551	7147	4460	8	2327	7157	4660	12	2537	7150	4516	8
case32	2478	7147	4493	8	2551	7147	4460	8	2327	7157	4660	12	2537	7150	4516	8
case33	5732	22706	11979	70	5784	22706	11509	70	5167	22795	12135	136	5349	22709	12058	70

Port Descriptions

This section describes the details for each interface. In addition, detailed information about configuration and control registers is included.

The AXI VDMA signals are described in Table 2-6.

Table 2-6: **AXI VDMA I/O Signal Description**

Signal Name	Interface	Signal Type	Init Status	Description
s_axi_lite_aclk	Clock	I		AXI VDMA AXI4-Lite interface clock ***Note:*** All aclk inputs must be tied to the same clock source when AXI VDMA is configured for synchronous clock mode (C_PRMRY_IS_ACLK_ASYNC=0).
m_axi_sg_aclk	Clock	I		AXI VDMA Scatter Gather clock ***Note:*** All aclk inputs must be tied to the same clock source when AXI VDMA is configured for synchronous clock mode (C_PRMRY_IS_ACLK_ASYNC=0).
m_axi_mm2s_aclk	Clock	I		AXI VDMA MM2S clock ***Note:*** All aclk inputs must be tied to the same clock source when AXI VDMA is configured for synchronous clock mode (C_PRMRY_IS_ACLK_ASYNC=0).

Table 2-6: **AXI VDMA I/O Signal Description *(Cont'd)***

Signal Name	Interface	Signal Type	Init Status	Description
m_axi_s2mm_aclk	Clock	I		AXI VDMA S2MM clock ***Note:*** All aclk inputs must be tied to the same clock source when AXI VDMA is configured for synchronous clock mode (C_PRMRY_IS_ACLK_ASYNC=0).
m_axis_mm2s_aclk	Clock	I		AXI VDMA MM2S AXIS clock ***Note:*** All aclk inputs must be tied to the same clock source when AXI VDMA is configured for synchronous clock mode (C_PRMRY_IS_ACLK_ASYNC=0).
s_axis_s2mm_aclk	Clock	I		AXI VDMA S2MM AXIS clock ***Note:*** All aclk inputs must be tied to the same clock source when AXI VDMA is configured for synchronous clock mode (C_PRMRY_IS_ACLK_ASYNC=0).
axi_resetn	Reset	I		AXI VDMA Reset. Active-Low reset. When asserted low, resets entire AXI VDMA core. Must be synchronous to s_axi_lite_aclk and asserted for a minimum eight clock cycles.
mm2s_introut	Interrupt	O	0	Interrupt Out for Memory Map to Stream Channel
s2mm_introut	Interrupt	O	0	Interrupt Out for Stream to Memory Map Channel
Video Synchronization Interface Signals				
mm2s_fsync	Frame Sync	I		MM2S Frame Sync Input. When enabled, VDMA Operations begin on each falling edge of fsync. This port is only valid when the channel uses external frame sync. AXI VDMA expects this signal to be asserted for minimum of one m_axis_mm2s_aclk cycle.
mm2s_fsync_out	Frame Sync	O	0	MM2S Frame Sync Output. This signal asserts High for one m_axis_mm2s_aclk cycle with each frame boundary. This signals indicates to target video IP when a transfer of MM2S new frame data begins.
mm2s_prmtr_update	Frame Sync	O	0	MM2S Parameter Update. This signal indicates that new mm2s video parameters take effect on next frame. This signal is asserted for one m_axis_mm2s_aclk cycle coincident with mm2s_fsync_out.

Table 2-6: **AXI VDMA I/O Signal Description *(Cont'd)***

Signal Name	Interface	Signal Type	Init Status	Description
s2mm_fsync	Frame Sync	I		S2MM Frame Sync Input. When enabled, VDMA operations begin on each falling edge of fsync. This port is only valid when the channel uses external frame sync. AXI VDMA expects this signal to be asserted for a minimum of one s_axis_s2mm_aclk cycle.
s2mm_fsync_out	Frame Sync	O	0	S2MM Frame Sync Output. This signal asserts High for one s_axis_s2mm_aclk cycle with each frame boundary. Indicates when S2MM new frame data can be transferred to the S2MM channel by video IP.
s2mm_prmtr_update	Frame Sync	O	0	S2MM Parameter Update. This signal indicates that new s2mm video parameters take effect on next frame. This signal is asserted for one s_axis_s2mm_aclk cycle coincident with s2mm_fsync_out.
Genlock Interface Signals				
mm2s_frame_ptr_in((C_MM2S_GENLOCK_NUM_MASTERS*6)-1: 0)	Genlock	I		MM2S Frame Pointer Input. • In Genlock Slave mode, it specifies the next frame for MM2S to operate on based on its FRMDLY setting. • In Dynamic Genlock Slave mode, it specifies the next frame for MM2S to operate on. • In Dynamic Genlock Master mode, it specifies the current frame that slave is operating on. See C_MM2S_GENLOCK_MODE in Parameter Descriptions for more details on different Genlock modes.
mm2s_frame_ptr_out(5:0)	Genlock	O	zeros	MM2S Frame Pointer Output. • In Genlock Master mode, it specifies the next frame for the slave VDMA to operate on based on slave VDMA's FRMDLY setting. • In Dynamic Genlock Master mode, it specifies the next frame for slave VDMA to operate on. • In Dynamic Genlock Slave mode, it specifies the current frame that slave is operating on. See C_MM2S_GENLOCK_MODE in Parameter Descriptions for more details on different Genlock modes.

Table 2-6: **AXI VDMA I/O Signal Description *(Cont'd)***

Signal Name	Interface	Signal Type	Init Status	Description
s2mm_frame_ptr_in((C_S2MM_GENLOCK_NUM_MASTERS*6)-1: 0)	Genlock	I		S2MM Frame Pointer Input. • In Genlock Slave mode, it specifies the next frame for S2MM to operate on based on its FRMDLY setting. • In Dynamic Genlock Slave mode, it specifies the next frame for S2MM to operate on. • In Dynamic Genlock Master mode, it specifies the current frame that slave is operating on. See C_MM2S_GENLOCK_MODE in Parameter Descriptions for more details on different Genlock modes.
s2mm_frame_ptr_out(5:0)	Genlock	O	zeros	S2MM Frame Pointer Output. • In Genlock Master mode, it specifies the next frame for the slave VDMA to operate on based on slave VDMA's FRMDLY setting. • In Dynamic Genlock Master mode, it specifies the next frame for slave VDMA to operate on. • In Dynamic Genlock Slave mode, it specifies the current frame that slave is operating on. See C_S2MM_GENLOCK_MODE in Parameter Descriptions for more details on different Genlock modes.
Line Buffer interface Signals				
mm2s_buffer_empty	LineBuffer	O	1	MM2S Line Buffer Empty. Indicates that the MM2S line buffer contains no stored data elements.
mm2s_buffer_almost_empty	LineBuffer	O	1	MM2S Line Buffer Almost Empty. Indicates that the MM2S line buffer has MM2S_FRMSTORE bytes or less stored. When mm2s_buffer_empty asserts, mm2s_buffer_almost_empty remains asserted.
s2mm_buffer_full	LineBuffer	O	0	S2MM Line Buffer Full. Indicates that the S2MM line buffer has no more room to store data elements.
s2mm_buffer_almost_full	LineBuffer	O	0	S2MM Line Buffer Almost Full. Indicates that the S2MM line buffer has S2MM_FRMSTORE bytes or more. When s2mm_buffer_full asserts, s2mm_buffer_almost_full remains asserted.

Table 2-6: **AXI VDMA I/O Signal Description *(Cont'd)***

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

Table 2-6: **AXI VDMA I/O Signal Description *(Cont'd)***

Signal Name	Interface	Signal Type	Init Status	Description
s_axi_lite_arready	S_AXI_LITE	O	0	AXI4-Lite Read Address Channel Read Address Ready. Indicates DMA is ready to accept the read address. • 1 = Ready to accept address. • 0 = Not ready to accept address.
s_axi_lite_araddr(31:0)	S_AXI_LITE	I		AXI4-Lite Read Address Bus.
s_axi_lite_rvalid	S_AXI_LITE	O	0	AXI4-Lite Read Data Channel Read Data Valid. • 1 = Read data is valid. • 0 = Read data is not valid.
s_axi_lite_rready	S_AXI_LITE	I		AXI4-Lite Read Data Channel Read Data Ready. Indicates target is ready to accept the read data. • 1 = Ready to accept data. • 0 = Not ready to accept data.
s_axi_lite_rdata(31:0)	S_AXI_LITE	O	Don't care	AXI4-Lite Read Data Bus
s_axi_lite_rresp(1:0)	S_AXI_LITE	O	Don't care	AXI4-Lite Read Response Channel Response. Indicates results of the read transfer. The AXI VDMA Lite interface always responds with OKAY. • 00b = OKAY — Normal access has been successful. • 01b = EXOKAY — Not supported. • 10b = SLVERR — Not supported. • 11b = DECERR — Not supported.
MM2S Memory Map Read Interface Signals				
m_axi_mm2s_araddr (C_M_AXI_MM2S_ADDR_ WIDTH-1: 0)	M_AXI_MM2S	O	Don't care	Read Address Channel Address Bus
m_axi_mm2s_arlen(7:0)	M_AXI_MM2S	O	Don't care	Read Address Channel Burst Length. In data beats - 1.
m_axi_mm2s_arsize(2:0)	M_AXI_MM2S	O	Don't care	Read Address Channel Burst Size. Indicates width of burst transfer. • 000b = 1 byte (8-bit wide burst). • 001b = 2 bytes (16-bit wide burst). • 010b = 4 bytes (32-bit wide burst). • 011b = 8 bytes (64-bit wide burst). • 100b = 16 bytes (128-bit wide burst). • 101b = 32 bytes (256-bit wide burst). • 110b = 64 bytes (512 bit wide burst). • 111b = 128 bytes (1024 bit wide burst).

Table 2-6: **AXI VDMA I/O Signal Description *(Cont'd)***

Signal Name	Interface	Signal Type	Init Status	Description
m_axi_mm2s_arburst(1:0)	M_AXI_MM2S	O	Don't care	Read Address Channel Burst Type. Indicates type burst. • 00b = FIXED — Not supported. • 01b = INCR — Incrementing address. • 10b = WRAP — Not supported. • 11b = Reserved.
m_axi_mm2s_arprot(2:0)	M_AXI_MM2S	O	000b	Read Address Channel Protection. Always driven with a constant output of 000b along with m_axi_mm2s_arvalid.
m_axi_mm2s_arcache(3:0)	M_AXI_MM2S	O	0011b	Read Address Channel Cache. Always driven with a constant output of 0011b along with m_axi_mm2s_arvalid.
m_axi_mm2s_arvalid	M_AXI_MM2S	O	0	Read Address Channel Read Address Valid. Indicates m_axi_mm2s_araddr is valid. • 1 = Read address is valid. • 0 = Read address is not valid.
m_axi_mm2s_arready	M_AXI_MM2S	I		Read Address Channel Read Address Ready. Indicates target is ready to accept the read address. • 1 = Target ready to accept address. • 0 = Target not ready to accept address.
m_axi_mm2s_rdata (C_M_AXI_MM2S_DATA_ WIDTH-1: 0)	M_AXI_MM2S	I		Read Data Channel Read Data.
m_axi_mm2s_rresp(1:0)	M_AXI_MM2S	I		Read Data Channel Response. Indicates results of the read transfer. • 00b = OKAY — Normal access has been successful. • 01b = EXOKAY — Not supported. • 10b = SLVERR — Slave returned error on transfer. • 11b = DECERR — Decode error, transfer targeted unmapped address.
m_axi_mm2s_rlast	M_AXI_MM2S	I		Read Data Channel Last. Indicates the last data beat of a burst transfer. • 1 = Last data beat. • 0 = Not last data beat.
m_axi_mm2s_rvalid	M_AXI_MM2S	I		Read Data Channel Data Valid. Indicates m_axi_mm2s_rdata is valid. • 1 = Valid read data. • 0 = Not valid read data.

Table 2-6: **AXI VDMA I/O Signal Description *(Cont'd)***

Signal Name	Interface	Signal Type	Init Status	Description
m_axi_mm2s_rready	M_AXI_MM2S	O		Read Data Channel Ready. Indicates the read channel is ready to accept read data. • 1 = Ready. • 0 = Not ready.
MM2S Master Stream Interface Signals				
mm2s_prmry_reset_out_n	M_AXIS_MM2S	O	0	Primary MM2S Reset Out.
m_axis_mm2s_tdata (C_M_AXIS_MM2S_TDATA_ WIDTH-1: 0)	M_AXIS_MM2S	O	Don't care	AXI4-Stream Data Out.
m_axis_mm2s_tkeep (C_M_AXIS_MM2S_TDATA_ WIDTH/8-1: 0)	M_AXIS_MM2S	O	Don't care	AXI4-Stream Write Keep. Indicates valid bytes on stream data. (For most use cases, all bytes will be valid.) • 1 = Byte is valid • 0 = Byte is not valid
m_axis_mm2s_tuser[C_M_A XIS_MM2S_TUSER_BITS-1:0]	M_AXIS_MM2S	O	Don't care	AXI4-Stream user bits. tuser(0) drives out mm2s start of frame (SOF). This signal is asserted for one clock period.
m_axis_mm2s_tvalid	M_AXIS_MM2S	O	0	AXI4-Stream Valid Out. Indicates stream data bus, m_axis_mm2s_tdata, is valid • 1 = Write data is valid. • 0 = Write data is not valid.
m_axis_mm2s_tready	M_AXIS_MM2S	I		AXI4-Stream Ready. Indicates to S2MM channel target is ready to receive stream data. • 1 = Ready to receive data. • 0 = Not ready to receive data.
m_axis_mm2s_tlast	M_AXIS_MM2S	O	Don't care	AXI4-Stream Last. Indicates last data beat of stream data. • 1 = Last data beat. • 0 = Not last data beat. See the Video IP: AXI Feature Adoption section of the *AXI Reference Guide* (UG76) for additional information.
S2MM Memory Map Write Interface Signals				
m_axi_s2mm_awaddr (C_M_AXI_S2MM_ADDR_ WIDTH-1: 0)	M_AXI_S2MM	O	Don't care	Write Address Channel Address Bus.
m_axi_s2mm_awlen(7: 0)	M_AXI_S2MM	O	Don't care	Write Address Channel Burst Length. In data beats - 1.

Table 2-6: **AXI VDMA I/O Signal Description *(Cont'd)***

Signal Name	Interface	Signal Type	Init Status	Description
m_axi_s2mm_awsize(2: 0)	M_AXI_S2MM	O	Don't care	Write Address Channel Burst Size. Indicates width of burst transfer. • 000b = 1 byte (8 bit wide burst). • 001b = 2 bytes (16 bit wide burst). • 010b = 4 bytes (32 bit wide burst). • 011b = 8 bytes (64 bit wide burst). • 100b = 16 bytes (128 bit wide burst). • 101b = 32 bytes (256 bit wide burst). • 110b = 64 bytes (512 bit wide burst). • 111b = 128 bytes (1024 bit wide burst).
m_axi_s2mm_awburst(1:0)	M_AXI_S2MM	O	Don't care	Write Address Channel Burst Type. Indicates type burst. • 00b = FIXED — Not supported. • 01b = INCR — Incrementing address. • 10b = WRAP — Not supported. • 11b = Reserved.
m_axi_s2mm_awprot(2:0)	M_AXI_S2MM	O	000b	Write Address Channel Protection. Always driven with a constant output of 000b along with m_axi_s2mm_awvalid.
m_axi_s2mm_awcache(3:0)	M_AXI_S2MM	O	0011b	Write Address Channel Cache. Always driven with a constant output of 0011b along with m_axi_s2mm_awvalid.
m_axi_s2mm_awvalid	M_AXI_S2MM	O	0	Write Address Channel Write Address Valid. Indicates if m_axi_s2mm_awaddr is valid. • 1 = Write Address is valid. • 0 = Write Address is not valid.
m_axi_s2mm_awready	M_AXI_S2MM	I		Write Address Channel Write Address Ready. Indicates target is ready to accept the write address. • 1 = Target read to accept address. • 0 = Target not ready to accept address.
m_axi_s2mm_wdata (C_M_AXI_S2MM_DATA_ WIDTH-1: 0)	M_AXI_S2MM	O	Don't care	Write Data Channel Write Data Bus.
m_axi_s2mm_wstrb (C_M_AXI_S2MM_DATA_ WIDTH/8 - 1: 0)	M_AXI_S2MM	O	Don't care	Write Data Channel Write Strobe Bus. Indicates which bytes are valid in the write data bus. This value is passed from the stream side strobe bus.
m_axi_s2mm_wlast	M_AXI_S2MM	O	Don't care	Write Data Channel Last. Indicates the last data beat of a burst transfer. • 1 = Last data beat. • 0 = Not last data beat.

Table 2-6: **AXI VDMA I/O Signal Description *(Cont'd)***

Signal Name	Interface	Signal Type	Init Status	Description
m_axi_s2mm_wvalid	M_AXI_S2MM	O	0	Write Data Channel Data Valid. Indicates m_axi_s2mm_wdata is valid. • 1 = Valid write data. • 0 = Not valid write data.
m_axi_s2mm_wready	M_AXI_S2MM	I		Write Data Channel Ready. Indicates the write channel target is ready to accept write data. • 1 = Target is ready • 0 = Target is not ready
m_axi_s2mm_bresp(1:0)	M_AXI_S2MM	I		Write Response Channel Response. Indicates results of the write transfer. • 00b = OKAY — Normal access has been successful. • 01b = EXOKAY — Not supported. • 10b = SLVERR — Slave returned error on transfer. • 11b = DECERR — Decode error, transfer targeted unmapped address.
m_axi_s2mm_bvalid	M_AXI_S2MM	I		Write Response Channel Response Valid. Indicates response, m_axi_s2mm_bresp, is valid. • 1 = Response is valid. • 0 = Response is not valid.
m_axi_s2mm_bready	M_AXI_S2MM	O	0	Write Response Channel Ready. Indicates MM2S write channel is ready to receive response. • 1 = Ready to receive response. • 0 = Not ready to receive response.
S2MM Slave Stream Interface Signals				
s2mm_prmry_reset_out_n	M_AXIS_S2MM	O	O	Primary S2MM Reset Out
s_axis_s2mm_tdata (C_S_AXIS_S2MM_TDATA_ WIDTH-1: 0)	S_AXIS_S2MM	I		AXI4-Stream Data In
s_axis_s2mm_tkeep (C_S_AXIS_S2MM_TDATA_ WIDTH/8-1: 0)	S_AXIS_S2MM	I		AXI4-Stream Write Keep. Indicates valid bytes on stream data. (For most use cases, all bytes are valid.). It needs to be tied High if stream master does not have this signal. • 1 = Byte is valid • 0 = Byte is not valid
s_axis_s2mm_tuser[C_S_ AXIS_S2MM_TUSER_BITS-1: 0]	M_AXIS_S2MM	I	Don't care	AXI4-Stream user bits. The signal tuser(0) receives in s2mm start of frame (SOF). AXI VDMA expects this signal to be asserted for one clock period only.

Table 2-6: **AXI VDMA I/O Signal Description *(Cont'd)***

Signal Name	Interface	Signal Type	Init Status	Description
s_axis_s2mm_tvalid	S_AXIS_S2MM	I		AXI4-Stream Valid In. Indicates stream data bus, s_axis_s2mm_tdata, is valid. • 1 = Write data is valid. • 0 = Write data is not valid.
s_axis_s2mm_tready	S_AXIS_S2MM	O	0	AXI4-Stream Ready. Indicates MM2S channel stream interface ready to receive stream data. • 1 = Ready to receive data. • 0 = Not ready to receive data.
s_axis_s2mm_tlast	S_AXIS_S2MM	I		AXI4-Stream Last. Indicates last data beat of stream data. • 1 = Last data beat. • 0 = Not last data beat. For additional information, see the Video IP: AXI Feature Adoption section of the UG76 *AXI Reference Guide*.
Scatter Gather Memory Map Read Interface Signals				
m_axi_sg_araddr (C_M_AXI_SG_ADDR_ WIDTH-1: 0)	M_AXI_SG	O	Don't care	Scatter Gather Read Address Channel Address Bus.
m_axi_sg_arlen(7: 0)	M_AXI_SG	O	Don't care	Scatter Gather Read Address Channel Burst Length. Length in data beats - 1.
m_axi_sg_arsize(2: 0)	M_AXI_SG	O	Don't care	Scatter Gather Read Address Channel Burst Size. Indicates width of burst transfer. • 000b = Not Supported by AXI VDMA SG Engine. • 001b = Not Supported by AXI VDMA SG Engine. • 010b = 4 bytes (32 bit wide burst). • 011b = Not Supported by AXI VDMA SG Engine. • 100b = Not Supported by AXI VDMA SG Engine. • 101b = Not Supported by AXI VDMA SG Engine. • 110b = Not Supported by AXI VDMA SG Engine. • 111b = Not Supported by AXI VDMA SG Engine.
m_axi_sg_arburst(1:0)	M_AXI_SG	O	Don't care	Scatter Gather Read Address Channel Burst Type. Indicates type burst. • 00b = FIXED — Not supported. • 01b = INCR — Incrementing address. • 10b = WRAP — Not supported. • 11b = Reserved.

Table 2-6: **AXI VDMA I/O Signal Description *(Cont'd)***

Signal Name	Interface	Signal Type	Init Status	Description
m_axi_sg_arprot(2:0)	M_AXI_SG	O	000b	Scatter Gather Read Address Channel Protection. Always driven with a constant output of 000b along with m_axi_sg_arvalid.
m_axi_sg_arcache(3:0)	M_AXI_SG	O	0011b	Scatter Gather Read Address Channel Cache. Always driven with a constant output of 0011b along with m_axi_sg_arvalid.
m_axi_sg_arvalid	M_AXI_SG	O	0	Scatter Gather Read Address Channel Read Address Valid. Indicates if m_axi_sg_araddr is valid. • 1 = Read Address is valid. • 0 = Read Address is not valid.
m_axi_sg_arready	M_AXI_SG	I		Scatter Gather Read Address Channel Read Address Ready. Indicates target is ready to accept the read address. • 1 = Target ready to accept address. • 0 = Target not ready to accept address.
m_axi_sg_rdata (C_M_AXI_SG_DATA_ WIDTH-1: 0)	M_AXI_SG	I		Scatter Gather Read Data Channel Read Data.
m_axi_sg_rresp(1:0)	M_AXI_SG	I		Scatter Gather Read Data Channel Response. Indicates results of the read transfer. • 00b = OKAY — Normal access has been successful. • 01b = EXOKAY — Not supported. • 10b = SLVERR — Slave returned error on transfer. • 11b = DECERR — Decode error, transfer targeted unmapped address.
m_axi_sg_rlast	M_AXI_SG	I		Scatter Gather Read Data Channel Last. Indicates the last data beat of a burst transfer. • 1 = Last data beat. • 0 = Not last data beat.
m_axi_sg_rvalid	M_AXI_SG	I		Scatter Gather Read Data Channel Data Valid. Indicates m_sg_aximry_rdata is valid. • 1 = Valid read data. • 0 = Not valid read data.
m_axi_sg_rready	M_AXI_SG	O	0	Scatter Gather Read Data Channel Ready. Indicates the read channel is ready to accept read data. • 1 = Is ready. • 0 = Is not ready.

Register Space

The AXI VDMA core register space for Register Direct mode is shown in Table 2-7 and for Scatter Gather Mode is shown in Table 2-8. The AXI VDMA Registers are memory-mapped into non-cacheable memory space. This memory space must be aligned on a AXI word (32-bit) boundary.

Endianess

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

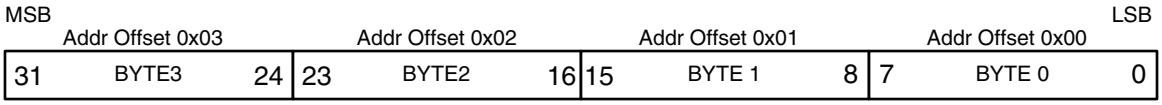


Figure 2-2: **32-bit Little Endian Example**

AXI VDMA Register Address Mapping For Register Direct Mode

Table 2-7: **Register Address Mapping for Register Direct Mode**

Address Space Offset (1)	Name	Description
00h	MM2S_DMACR	MM2S DMA Control Register
04h	MM2S_DMASR	MM2S DMA Status Register
08 to 10h	Reserved	N/A
14h	MM2S_REG_INDEX	MM2S Register Index
18h	MM2S_FRMSTORE	MM2S Frame Store Register
1Ch	MM2S_THRESHOLD	MM2S Line Buffer Threshold Register
20h	Reserved	N/A
24h	FRMPTR_STS	MM2S and S2MM Current Frame Pointer Status
28h	PARK_PTR_REG	MM2S and S2MM Park Pointer Register
2Ch	VDMA_VERSION	Video DMA Version Register
30h	S2MM_DMACR	S2MM DMA Control Register
34h	S2MM_DMASR	S2MM DMA Status Register
38h to 40h	Reserved	N/A
44h	S2MM_REG_INDEX	S2MM Register Index
48h	S2MM_FRMSTORE	S2MM Frame Store Register

Table 2-7: **Register Address Mapping for Register Direct Mode *(Cont'd)***

Address Space Offset [1]	Name	Description
4Ch	S2MM_THRESHOLD	S2MM Line Buffer Threshold Register
50h	MM2S_VSIZE[3]	MM2S Vertical Size Register
54h	MM2S_HSIZE[3]	MM2S Horizontal Size Register
58h	MM2S_FRMDLY_STRIDE[3]	MM2S Frame Delay and Stride Register
5Ch	MM2S_START_ADDRESS1[3]	MM2S Start Address 1
60h	MM2S_START_ADDRESS2[2] [3]	MM2S Start Address 2
64h	MM2S_START_ADDRESS3[2] [3]	MM2S Start Address 3
68h	MM2S_START_ADDRESS4[2] [3]	MM2S Start Address 4
6Ch	MM2S_START_ADDRESS5[2][3]	MM2S Start Address 5
70h	MM2S_START_ADDRESS6[2] [3]	MM2S Start Address 6
74h	MM2S_START_ADDRESS7[2] [3]	MM2S Start Address 7
78h	MM2S_START_ADDRESS8[2] [3]	MM2S Start Address 8
7Ch	MM2S_START_ADDRESS9[2] [3]	MM2S Start Address 9
80h	MM2S_START_ADDRESS10[2] [3]	MM2S Start Address 10
84h	MM2S_START_ADDRESS11[2] [3]	MM2S Start Address 11
88h	MM2S_START_ADDRESS12[2] [3]	MM2S Start Address 12
8Ch	MM2S_START_ADDRESS13[2] [3]	MM2S Start Address 13
90h	MM2S_START_ADDRESS14[2] [3]	MM2S Start Address 14
94h	MM2S_START_ADDRESS15[2] [3]	MM2S Start Address 15
98h	MM2S_START_ADDRESS16[2] [3]	MM2S Start Address 16
9Ch	Reserved	N/A
A0h	S2MM_VSIZE[3]	S2MM Vertical Size Register
A4h	S2MM_HSIZE[3]	S2MM Horizontal Size Register
A8h	S2MM_FRMDLY_STRIDE[3]	S2MM Frame Delay and Stride Register
ACh	S2MM_START_ADDRESS1[3]	S2MM Start Address 1
B0h	S2MM_START_ADDRESS2[2] [3]	S2MM Start Address 2
B4h	S2MM_START_ADDRESS3[2] [3]	S2MM Start Address 3
B8h	S2MM_START_ADDRESS4[2] [3]	S2MM Start Address 4
BCh	S2MM_START_ADDRESS5[2] [3]	S2MM Start Address 5
C0h	S2MM_START_ADDRESS6[2] [3]	S2MM Start Address 6
C4h	S2MM_START_ADDRESS7[2] [3]	S2MM Start Address 7
C8h	S2MM_START_ADDRESS8[2] [3]	S2MM Start Address 8
CCh	S2MM_START_ADDRESS9[2][3]	S2MM Start Address 9
D0h	S2MM_START_ADDRESS10[2] [3]	S2MM Start Address 10
D4h	S2MM_START_ADDRESS11[2][3]	S2MM Start Address 11

Table 2-7: **Register Address Mapping for Register Direct Mode *(Cont'd)***

Address Space Offset [(1)]	Name	Description
D8h	S2MM_START_ADDRESS12[(2) (3)]	S2MM Start Address 12
DCh	S2MM_START_ADDRESS13[(2) (3)]	S2MM Start Address 13
E0h	S2MM_START_ADDRESS14[(2)(3)]	S2MM Start Address 14
E4h	S2MM_START_ADDRESS15[(2) (3)]	S2MM Start Address 15
E8h	S2MM_START_ADDRESS16[(2) (3)]	S2MM Start Address 16
ECh	Reserved	N/A
F0h	S2MM_HSIZE_STATUS	S2MM hsize status Register
F4h	S2MM_VSIZE_STATUS	S2MM vsize status Register

1. Address Space Offset is relative to C_BASEADDR assignment.
2. Start Addresses 2 to 32 for MM2S and S2MM depend on C_NUM_FSTORES parameter. Start address registers greater than C_NUM_FSTORES setting are reserved. Only MM2S_FRMSTORE or S2MM_FRMSTORE start address registers for the respective channel are used for transfers. See the MM2S_REG_INDEX and S2MM_REG_INDEX register definitions for accessing 32 start address registers.
3. Video parameter and start address registers are Read/Writable when the video parameter reads are enabled. (C_ENABLE_VIDPRMTR_READS=1) and are Write Only when the video parameter reads are disabled. (C_ENABLE_VIDPRMTR_READS=0).

AXI VDMA Register Address Mapping For Scatter Gather Mode

Table 2-8: **Register Address Mapping for Scatter Gather Mode**

Address Space Offset[a]	Name	Description
00h	MM2S_DMACR	MM2S DMA Control Register
04h	MM2S_DMASR	MM2S DMA Status Register
08h	MM2S_CURDESC	MM2S Current Descriptor Pointer
0Ch	Reserved	N/A
10h	MM2S_TAILDESC	MM2S Tail Descriptor Pointer
14h	Reserved	N/A
18h	MM2S_FRMSTORE	MM2S Frame Store Register
1Ch	MM2S_THRESHOLD	MM2S Line Buffer Threshold Register
20h	Reserved	N/A
24h	FRMPTR_STS	MM2S and S2MM Current Frame Pointer Status
28h	PARK_PTR_REG	MM2S and S2MM Park Pointer Register
2Ch	VDMA_VERSION	Video DMA Version Register
30h	S2MM_DMACR	S2MM DMA Control Register
34h	S2MM_DMASR	S2MM DMA Status Register
38h	S2MM_CURDESC	S2MM Current Descriptor Pointer

Table 2-8: **Register Address Mapping for Scatter Gather Mode *(Cont'd)***

Address Space Offset[a]	Name	Description
3Ch	Reserved	N/A
40h	S2MM_TAILDESC	S2MM Tail Descriptor Pointer
44h	Reserved	N/A
48h	S2MM_FRMSTORE	S2MM Frame Store Register
4Ch	S2MM_THRESHOLD	S2MM Line Buffer Threshold Register
50h - EFh	Reserved	N/A
F0h	S2MM_HSIZE_STATUS	S2MM hsize status Register
F4h	S2MM_VSIZE_STATUS	S2MM vsize status Register

a. Address Space Offset is relative to C_BASEADDR assignment. C_BASEADDR is defined in AXI VDMA mpd file and set by XPS.

Memory Map to Stream Register Detail

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

This register provides control for the Memory Map to Stream DMA Channel for both Scatter Gather mode and Register Direct mode.

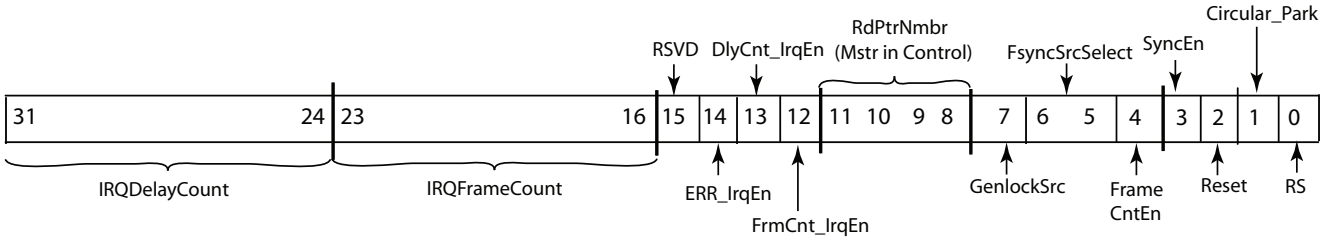


Figure 2-3: **MM2S DMACR Register**

Table 2-9: **MM2S_DMACR Register Details**

Bits	Field Name	Default Value	Access Type	Description
31 downto 24	IRQDelayCount	00h	R/W	This value is used for setting the interrupt delay count value. The delay count interrupt is a mechanism for causing the DMA engine to generate an interrupt after the delay period has expired. Timer begins counting either upon receipt of frame sync (if C_USE_FSYNC=1,2,3) or completion of vsize lines (if C_USE_FSYNC=0). It resets with subsequent start of packet (m_axis_mm2s_tvalid) assertion. When a value different than the current IRQDelayCount is written to this field, the internal delay counter is reset to the new value. Setting this value to zero disables the delay counter interrupt.
23 downto 16	IRQFrameCount	01h	R/W	This value is used for setting the interrupt threshold. When frame transfer interrupt events occur, an internal counter counts down from the Interrupt Frame Count setting. When the count reaches zero, an interrupt out is generated by the VDMA engine. When a value different than the current IRQFrameCount is written to this field, the internal frame counter is reset to the new value. The minimum setting for the count is 0x01. A write of 0x00 to this register sets the count to 0x01. When DMACR.FrameCntEn = 1, this value determines the number of frame buffers to process.
15	Reserved	0	RO	Writing to this bit has no effect and it is always read as zeros.
14	Err_IrqEn	0	R/W	Interrupt on Error Interrupt Enable. When set to 1, allows DMASR.Err_Irq to generate an interrupt out. 0 = Error Interrupt disabled 1 = Error Interrupt enabled
13	DlyCnt_IrqEn	0	R/W	Interrupt on Delay Count Interrupt Enable. When set to 1, allows DMASR.DlyCnt_Irq to generate an interrupt out. 0 = Delay Count Interrupt disabled 1 = Delay Count Interrupt enabled
12	FrmCnt_IrqEn	0	R/W	Frame Count Complete Interrupt Enable. When set to 1, allows DMASR.FrmCnt_Irq to generate an interrupt out when IRQFrameCount value reaches zero. 0 = Frame Count Interrupt disabled 1 = Frame Count Interrupt enabled

Table 2-9: **MM2S_DMACR Register Details *(Cont'd)***

Bits	Field Name	Default Value	Access Type	Description
11 downto 8	RdPntrNum	zeros	R/W	Indicates the master in control when MM2S channel is configured for Genlock slave/Dynamic Genlock Master/Dynamic Genlock Slave (C_MM2S_GENLOCK_MODE = 1,2,3). 0000b = Controlling entity is Entity 1 0001b = Controller entity is Entity 2 0010b = Controller entity is Entity 3 and so on. Maximum valid RdPntrNum is C_MM2S_GENLOCK_NUM_MASTER - 1. Setting to a value greater than C_MM2S_GENLOCK_NUM_MASTER - 1 has undefined results.
7	GenlockSrc	0	R/W	Sets the Genlock source for Genlock slaves. • 0 = External Genlock • 1 = Internal Genlock This bit has meaning only: • if both VDMA channels are enabled AND • if one VDMA channel is configured as Genlock Master then the other VDMA channel *must* be configured as Genlock Slave OR if one VDMA channel is configured as Dynamic Genlock Master then the other VDMA channel *must* be configured as Dynamic Genlock Slave AND • if C_INCLUDE_INTERNAL_GENLOCK = 1 See C_MM2S_GENLOCK_MODE in Parameter Descriptions for more details on different Genlock modes.
6 downto 5	FsyncSrcSelect	00	R/W	Selects the frame sync source for the MM2S channel. The frame sync source is selected as follows: • 00 = mm2s_fsync • 01 = s2mm_fsync • 10 = reserved • 11 = reserved ***Note:*** Frame Sync Source Select is only valid if configured for external frame sync.
4	FrameCntEn	0	R/W	Configures the MM2S channel to allow only IRQFrameCount number of transfers to occur. After IRQFrameCount frames have been transferred, the MM2S channel halts, DMACR.RS bit is cleared to 0, and DMASR.Halted asserts to 1 when the channel has completely halted.

Table 2-9: **MM2S_DMACR Register Details *(Cont'd)***

Bits	Field Name	Default Value	Access Type	Description
3	SyncEn	0	R/W	Enables Genlock or Dynamic Genlock Synchronization. • 0 = Genlock or Dynamic Genlock Synchronization disabled. Genlock input is ignored by MM2S. • 1 = Genlock or Dynamic Genlock Synchronization enabled. MM2S synchronized to Genlock frame input. ***Note:*** This value is only valid when the channel is configured as Genlock Slave or Dynamic Genlock Master or Dynamic Genlock Slave(C_MM2S_GENLOCK_MODE = 1 or 2 or 3). If configured for Genlock Master mode (C_MM2S_GENLOCK_MODE = 0), this bit is reserved and always reads as zero. See C_MM2S_GENLOCK_MODE in Parameter Descriptions for more details on different Genlock modes.
2	Reset	0	R/W	Soft reset for resetting the AXI VDMA MM2S channel. Setting this bit to a 1 causes the AXI VDMA MM2S channel to be reset. Reset is accomplished gracefully. Pending commands/transfers are flushed or completed. AXI4-Stream reset output is asserted. Setting DMACR.Reset = 1 only resets the MM2S channel. After completion of a soft reset all MM2S registers and bits are in the Reset State. 0 = Reset NOT in progress — Normal operation 1 = Reset in progress
1	Circular_Park	1	R/W	Indicates frame buffer Circular mode or frame buffer Park mode. 0 = Park Mode— Engine will park on frame buffer referenced by PARK_PTR_REG.RdFrmPntrRef. 1 = Circular Mode — Engine continuously circles through MM2S_FRMSTORE frame buffers.

Table 2-9: **MM2S_DMACR Register Details *(Cont'd)***

Bits	Field Name	Default Value	Access Type	Description
0	RS	0	R/W	Run / Stop controls running and stopping of the VDMA channel. For any DMA operations to commence, the AXI VDMA engine must be running (DMACR.RS=1). 0 = Stop ─ VDMA stops when current (if any) DMA operations are complete. Fetched descriptors are flushed inside VDMA. The halted bit in the DMA Status Register asserts to 1 when the DMA engine is halted. This bit gets cleared by AXI VDMA hardware when an error occurs or when the IRQFrameCount is reached when Frame Count Enable is asserted (DMACR.FrameCntEn = 1). The CPU can also choose to clear this bit to stop DMA operations. 1 = Run ─ Start DMA operations. The halted bit in the DMA Status Register deasserts to 0 when the DMA engine begins operations. ***Note:*** On Run/Stop clear, in-progress stream transfers might terminate early.
RO = Read Only. Writing has no effect. R/W = Read / Write.				

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

This register provides the status for the Memory Map to Stream DMA Channel for both Scatter Gather mode and Register Direct mode.

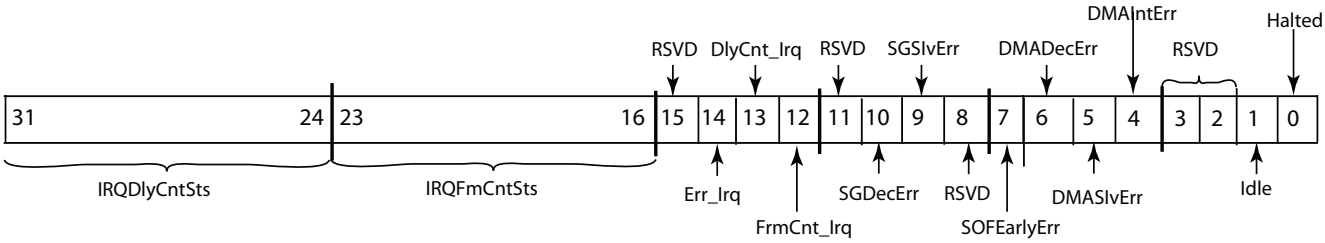


Figure 2-4: **MM2S DMASR Register**

Table 2-10: **MM2S_DMASR Register Details**

Bits	Field Name	Default Value	Access Type	Description
31 downto 24	IRQDelayCntSts	00h	RO	Interrupt Delay Count Status. Indicates current interrupt delay time value.
23 downto 16	IRQFrameCntSts	01h	RO	Interrupt Frame Count Status. Indicates current interrupt frame count value.

Table 2-10: **MM2S_DMASR Register Details *(Cont'd)***

Bits	Field Name	Default Value	Access Type	Description
15	Reserved	0	RO	Always read as zero.
14	Err_Irq	0	R/WC	Interrupt on Error. • 0 = No error Interrupt. • 1 = Error interrupt detected. If enabled (DMACR.Err_IrqEn = 1), an interrupt out is generated when error is detected.
13	DlyCnt_Irq	0	R/WC	Interrupt on Delay. • 0 = No Delay Interrupt. • 1 = Delay Interrupt detected. If enabled (DMACR.DlyCnt_IrqEn = 1), an interrupt out is generated when delay count reaches its programmed value.
12	FrmCnt_Irq	0	R/WC	Frame Count Interrupt. • 0 = No Frame Count Interrupt. • 1 = Frame Count Interrupt detected. If enabled (DMACR.FrmCnt_IrqEn = 1) and if the interrupt threshold has been met, an interrupt out is generated from the AXI VDMA
11	Reserved	0	RO	Writing to this bit has no effect, and it is always read as zeros.
10	SGDecErr	0	RO	Scatter Gather Decode Error. • 0 = No SG Decode Errors. • 1 = SG Decode Error detected. DMA Engine halts. See Errors for more information.
9	SGSlvErr	0	RO	Scatter Gather Slave Error. • 0 = No SG Slave Errors. • 1 = SG Slave Error detected. DMA Engine halts. See Errors for more information.
8	Reserved	0	RO	Writing to this bit has no effect, and it is always read as zeros.
7	SOFEarlyErr	0	RO or R/WC	Start of Frame Early Error • 0 = No Start of Frame Error • 1 = Start of Frame Early Error detected This error occurs if mm2s_fsync is received before the completion of frame on streaming interface. ***Note:*** In Flush On Frame Sync mode, this bit is R/WC (Write 1 to Clear) bit. Otherwise it is a Read Only bit.

Table 2-10: **MM2S_DMASR Register Details *(Cont'd)***

Bits	Field Name	Default Value	Access Type	Description
6	DMADecErr	0	RO	DMA Decode Error. This error occurs if the address request is to an invalid address. • 0 = No DMA Decode Errors. • 1 = DMA Decode Error detected. DMA channel halts.
5	DMASlvErr	0	RO	DMA Slave Error. • 0 = No DMA Slave Errors. • 1 = DMA Slave Error detected. DMA Engine halts. This error occurs if the slave read from the Memory Map interface issues a Slave Error.
4	DMAIntErr	0	RO or R/WC	DMA Internal Error. • 0 = No DMA Internal Errors. • 1 = DMA Internal Error detected. DMA channel halts. This error occurs during one of the following conditions: • Descriptor is fetched with hsize or vsize = 0 in Scatter Gather mode • HSIZE or VSIZE register were written zeros in Register Direct mode • Transferred frame size is greater than vsize values ***Note:*** In Flush On Frame Sync mode, this bit is R/WC (Write 1 to Clear) bit. Otherwise its a Read Only bit.
3 downto 2	Reserved	0	RO	Writing to these bits has no effect, and they are always read as zeros.

Table 2-10: **MM2S_DMASR Register Details *(Cont'd)***

Bits	Field Name	Default Value	Access Type	Description
1	Idle	0	RO	DMA Scatter Gather Engine Idle. In Scatter Gather Mode (C_INCLUDE_SG = 1) this bit indicates the state of AXI VDMA Scatter Gather Engine operations. The assertion of Idle indicates the SG Engine has reached the tail pointer for the associated channel and all queued descriptors have been processed. If in the Idle state (DMASR.Idle = 1), writing to the TailPointer register automatically restarts DMA operations. For Register Direct Mode (C_INCLUDE_SG = 0) this bit is reserved and always read as 0b. • 0 = Not Idle — SG operations for MM2S channel in progress. • 1 = Idle — SG operation for MM2S channel paused. ***Note:*** DMASR.Idle only asserts after the SG engine has passed through the descriptor chain at least once and has reached the TAILDESC. ***Note:*** Writing to the TAILDESC register when not Idle (DMASR.Idle = 0) produces undefined results.
0	Halted	1	RO	DMA Channel Halted. DMA Channel Halted. Indicates the run/stop state of the DMA channel. • 0 = DMA channel running • 1 = DMA channel halted. This bit gets set when DMACR.RS = 0 and DMA and SG operations have halted. There can be a lag of time between when DMACR.RS = 0 and when DMASR.Halted = 1. ***Note:*** When halted (RS= 0 and Halted = 1), writing to CURDESC or TAILDESC pointer registers in Scatter Gather mode (C_INCLUDE_SG = 1) or Register Direct Mode (C_INCLUDE_SG = 0) has no effect on DMA operations.
RO = Read Only. Writing has no effect. R/WC = Read / Write to Clear. A CPU write of 1 clears the associated bit to 0.				

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

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

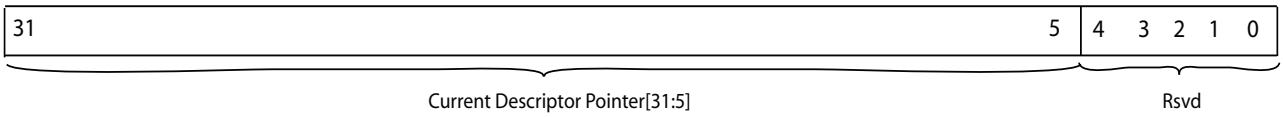


Figure 2-5: **MM2S CURDESC Register**

Table 2-11: **MM2S_CURDESC Register Details**

Bits	Field Name	Default Value	Access Type	Description
31 downto 5	Current Descriptor Pointer	zeros	R/W (RO)	In Scatter Gather Mode (C_INCLUDE_SG = 1) indicates the pointer of the current descriptor being worked on. This register must contain a pointer to a valid descriptor prior to writing the TAILDESC register. Otherwise, undefined results occur. When DMACR.RS is 1, CURDESC becomes Read Only (RO) and is used to fetch the first descriptor. When the DMA Engine is running (DMACR.RS=1), CURDESC registers are updated by AXI VDMA to indicate the current descriptor being worked on. On Scatter Gather error detection, CURDESC is updated to reflect the descriptor associated with the detected error. The register can only be written to by the CPU when the DMA Engine is Halted (DMACR.RS=0 and DMASR.Halted =1). At all other times, this register is Read Only (RO). Descriptors must be 8-word aligned, that is, 0x00, 0x20, 0x40, and so on. Any other alignment has undefined results. In Register Direct Mode (C_INCLUDE_SG = 0) this field is reserved and always read as zeros.
4 downto 0	Reserved	0	RO	Writing to these bits has no effect, and they are always read as zeros.
RO = Read Only. Writing has no effect. R/W = Read / Write.				

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

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

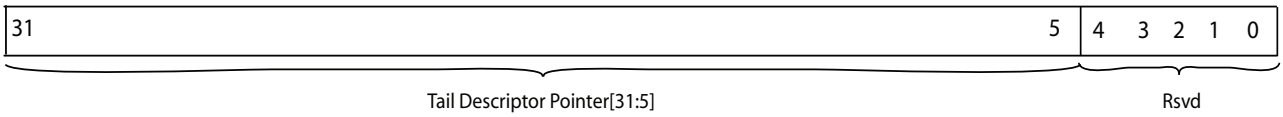


Figure 2-6: **MM2S_TAILDESC Register**

Table 2-12: **MM2S_TAILDESC Register Details**

Bits	Field Name	Default Value	Access Type	Description
31 downto 5	Tail Descriptor Pointer	zeros	R/W (RO)	In Scatter Gather Mode (C_INCLUDE_SG = 1), indicates the pause pointer in a descriptor chain. The AXI VDMA SG Engine pauses descriptor fetching after completing operations on the descriptor whose current descriptor pointer matches the tail descriptor pointer. When AXI VDMA Channel is not Halted (DMASR.Halted = 0), a write by the CPU to the TAILDESC register causes the AXI VDMA SG Engine to start fetching descriptors or restart if it was idle (DMASR.Idle = 1). Writing to the TAILDESC when not idle (DMASR.Idle = 0) has undefined results. If the AXI DMA Channel is Halted (DMASR.Halted = 1 and DMACR.RS = 0), a write by the CPU to the TAILDESC register has no effect except to reposition the pause point. ***Note:*** Descriptors must be 8-word aligned, that is, 0x00, 0x20, 0x40, and so on. Any other alignment has undefined results. ***Note:*** In Register Direct Mode (C_INCLUDE_SG = 0) this field is reserved and always read as zeros.
4 downto 0	Reserved	0	RO	Writing to these bits has no effect, and they are always read as zeros.
RO = Read Only. Writing has no effect. R/W = Read / Write				

MM2S_REG_INDEX (MM2S Register Index - Offset 14h) (C_INCLUDE_SG = 0)

This register provides access to upper bank of 16 (that is, 17 through 32) start addresses.

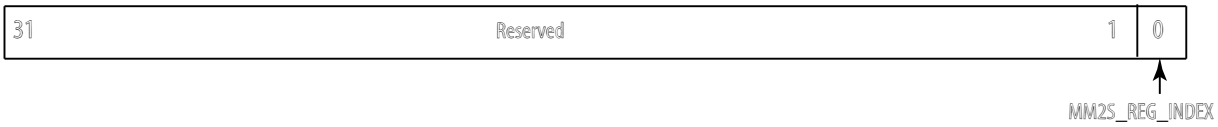


Figure 2-7: **MM2S Register Index**

Table 2-13: **MM2S Register Index (MM2S_REG_INDEX – Offset 0x14)**

Bits	Field Name	Default/ Reset State	Access	
31 downto 1		Reserved	RO	Always read as zero
0	MM2S Reg Index	zeroes	R/W	When set, enables access to the next set of 16 Frame Store Start Addresses (Bank1- 17 through 32) depending upon the following cases: Case 1: When C_NUM_FSTORES is less than or equal to 16, Bank1 (17 thru 32) registers are not available. Any writes to this bit do not change the behavior of VDMA. Case 2: When C_NUM_FSTORES is greater than 16 but less than 32 • 0 = Any write or read access between 0x5C to 0x98 accesses the Bank0 (1 thru 16) Frame StoreStart Address registers. • 1 = Accesses Bank1 registers. Example: If C_NUM_FSTORES = 20, Bank0 has 1-16 Frame Store Start Addresses and Bank1 has 17-20 Frame Store Start Addresses. Any access to Frame Store Start Addresses above 20 has no effect on writes and returns zero on reads. Case 3.: When C_NUM_FSTORES is equal to 32 • 0 = Any write or read access between 0x5C to 0x98 accesses the Bank0 registers. • 1 = Any write or read access between 0x5C to 0x98 accesses the Bank1 registers.
Note: MM2S_REG_INDEX register is not present in case of SG=1 mode. ***Note:*** The existing VDMA behavior of Dynamic MM2S Frame Store selection (MM2S) remains unchanged with the addition of the MM2S_REG_INDEX register.				

MM2S_FRMSTORE (MM2S Frame Store Register– Offset 18h) (C_INCLUDE_SG = 1/0)

This register provides the number of Frame Stores to use for the Memory Map to Stream channel.

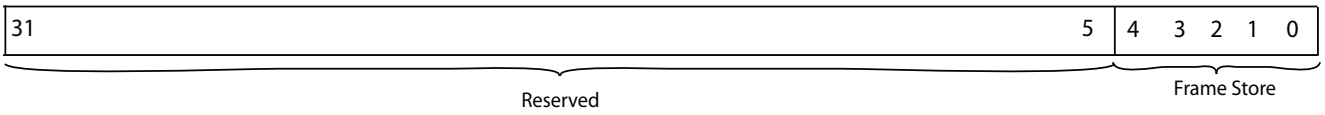


Figure 2-8: **MM2S_FRMSTORE Register**

Table 2-14: **MM2S_FRMSTORE Register Details**

Bits	Field Name	Default Value	Access Type	Description
31 downto 6	Reserved	zeroes	RO	Writing to these bits has no effect, and they are always read as zeros.
5 downto 0	Frame Store	C_NUM_FSTORES	R/W	Indicates the number of frame stores to use for video data transfers. This value defaults to C_NUM_FSTORES. For Scatter Gather mode (C_INCLUDE_SG = 1), this value specifies the number of Scatter Gather Descriptors required. For Register Direct mode (C_INCLUDE_SG = 0) this value specifies the number of Start Address registers used for transfers. On reset and start-up this register is set to C_NUM_FSTORES. ***Note:*** Genlock Masters and their attached Genlock Slaves must have identical Frame Store settings. Any mismatch in values has undefined results. This valid for internal genlock. ***Note:*** Dynamic Genlock Masters and their attached Dynamic Genlock Slaves must have identical Frame Store settings. Any mismatch in values has undefined results. ***Note:*** Values written must be greater than 0 and less than or equal to C_NUM_FSTORES. Any other value has undefined results.
RO = Read Only - Writing has no effect. R/W = Read / Write				

MM2S_THRESHOLD (MM2S Line Buffer Threshold Register– Offset 1Ch) (C_INCLUDE_SG = 1/0)

This register provides the Line Buffer Threshold for the Memory Map to Stream channel.

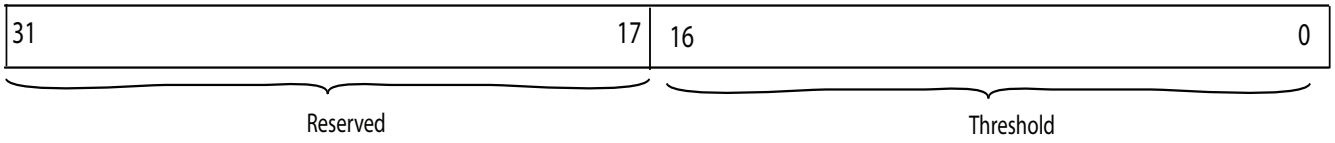


Figure 2-9: **MM2S_THRESHOLD Register**

Table 2-15: **MM2S_THRESHOLD Register Details**

Bits	Field Name	Default Value	Access Type	Description
31 downto 17	Reserved	zeroes	RO	Reserved. Always read as zero
16 downto 0	Line Buffer Threshold	C_MM2S_ LINEBUFFER _THRESH	R/W	Threshold point at which MM2S line buffer almost empty flag asserts high. Threshold specified in bytes and must be a multiple of C_M_AXIS_MM2S_TDATA_WIDTH/8, subject to the following condition. When Stream Data Width value is equal to a non-power of 2 value (that is, 24, 40, 72, 136, 264, 520), Threshold follows the restriction imposed by the next nearest upper power of 2 value (that is, 32, 64, 128, 256, 512, 1024 respectively). Stream Data Width — Allowed Values 8 — 1, 2, 3,..... 16 — 2, 4, 6,..... 24, 32 — 4, 8, 12,..... 40 to 64 — 8, 16, 24,..... 72 to 128 — 16, 32, 48,..... 136 to 256 — 32, 64, 96,.... 264 to 512 — 64, 128, 192,... 520 to 1024 — 128, 256, 384,.. ***Note:*** Maximum threshold value is limited by C_MM2S_LINEBUFFER_DEPTH. ***Note:*** Value valid when MM2S line buffer is included (C_MM2S_LINEBUFFER_DEPTH > 0).

RO = Read Only - Writing has no effect.
R/W = Read / Write

FRMPTR_STS (MM2S and S2MM Current Frame Pointer Status -- Offset 24h) (C_INCLUDE_SG = 1/0)

This register provides current operating frame pointer status of both MM2S and S2MM channels. This helps in tracking frame pointers when they are operating in different Genlock modes.

Table 2-16: **FRMPTR_STS Register Details**

Bits	Field Name	Default Value	Access Type	Description
31 downto 29	Reserved	0	RO	Writing to these bits has no effect, and they are always read as zeros.
28 downto 24	S2MMFrmPtrIn	0	RO	S2MM Frame Pointer Input • Reserved if S2MM channel is Genlock Master • Indicates Genlock Master's frame pointer value if S2MM channel is Genlock Slave • Indicates Dynamic Genlock Slave's frame pointer value if S2MM channel is Dynamic Genlock Master • Indicates Dynamic Genlock Master's frame pointer value if S2MM channel is Dynamic Genlock Slave
23 downto 21	Reserved	0	RO	Writing to these bits has no effect, and they are always read as zeros.
20 downto 16	S2MMFrmPtrOut	0	RO	Indicates current working frame of S2MM channel.
15 downto 13	Reserved	0	RO	Writing to these bits has no effect, and they are always read as zeros.
12 downto 8	MM2SFrmPtrIn	0	RO	MM2S Frame Pointer Input • Reserved if MM2S channel is Genlock Master • Indicates Genlock Master's frame pointer value if MM2S channel is Genlock Slave • Indicates Dynamic Genlock Slave's frame pointer value if MM2S channel is Dynamic Genlock Master • Indicates Dynamic Genlock Master's frame pointer value if MM2S channel is Dynamic Genlock Slave
7 downto 5	Reserved	0	RO	Writing to these bits has no effect, and they are always read as zeros.
4 downto 0	MM2SFrmPtrOut	0	RO	Indicates current working frame of MM2S channel.

PARK_PTR_REG (Park Pointer Register – Offset 28h) (C_INCLUDE_SG = 1/0)

This register provides Park Pointer Registers for the Memory Map to Stream and Stream to Memory Map DMA transfer Management.

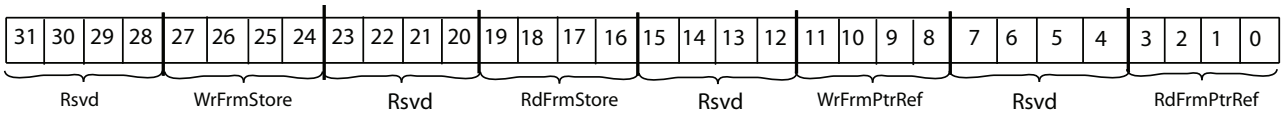


Figure 2-10: **PARK_PTR_REG Register**

Table 2-17: **PARK_PTR_REG Register Details**

Bits	Field Name	Default Value	Access Type	Description
31 downto 29	Reserved	0	RO	Writing to these bits has no effect, and they are always read as zeros.
28 downto 24	WrFrmStore		RO	Write Frame Store. Indicates current frame being operated on by S2MM channel. During DMA operations this value continually updates as each frame is processed. During error conditions, the value is updated with the current frame being operated on when the error occurred.
23 downto 21	Reserved	0	RO	Writing to these bits has no effect, and they are always read as zeros.
20 downto 16	RdFrmStore		RO	Read Frame Store. Indicates current frame being operated on by MM2S channel. During DMA operations this value continually updates as each frame is processed. During error conditions, the value is updated with the current frame being operated on when the error occurred.
15 downto 13	Reserved	0	RO	Writing to these bits has no effect, and they are always read as zeros.
12 downto 8	WrFrmPtrRef		R/W	Write Frame Pointer Reference. When Parked (DMACR.Circular_Park = 0), references the S2MM Frame to park on.
7 downto 5	Reserved	0	RO	Writing to these bits has no effect, and they are always read as zeros.
4 downto 0	RdFrmPtrRef		R/W	Read Frame Pointer Reference. When Parked (DMACR.Circular_Park = 0) references the MM2S Frame to park on.
RO = Read Only. Writing has no effect. R/W = Read / Write				

VDMA_VERSION (AXI VDMA Version Register – Offset 2Ch) (C_INCLUDE_SG = 1/0)

This register provides the AXI VDMA Version.

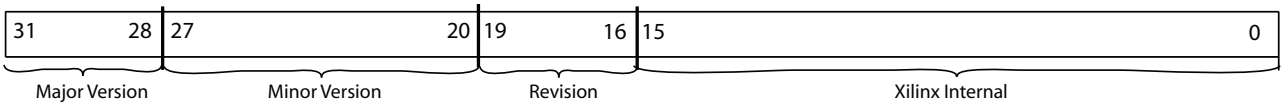


Figure 2-11: **VDMA_VERSION Register**

Table 2-18: **VDMA_VERSION Register Details**

Bits	Field Name	Default Value	Access Type	Description
31 downto 28	Major Version	5h	RO	Single 4-bit hexadecimal value. v1 = 1h, v2=2h, v3=3h, and so on.
27 downto 20	Minor Version	02h	RO	Two separate 4-bit hexadecimal values. 00 = 00h, 01 = 01h, and so on.
19 downto 16	Revision	Ah	RO	Revision letter as a hexadecimal character from 'a' to 'f'. Mapping is as follows: Ah-> 'a', Bh -> 'b', Ch-> 'c', and so on.
15 downto 0	Xilinx Internal	various	RO	Reserved for Internal Use Only. Integer value from 0 to 9999.
RO = Read Only. Writing has no effect.				

Stream to Memory Map Register Detail

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

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

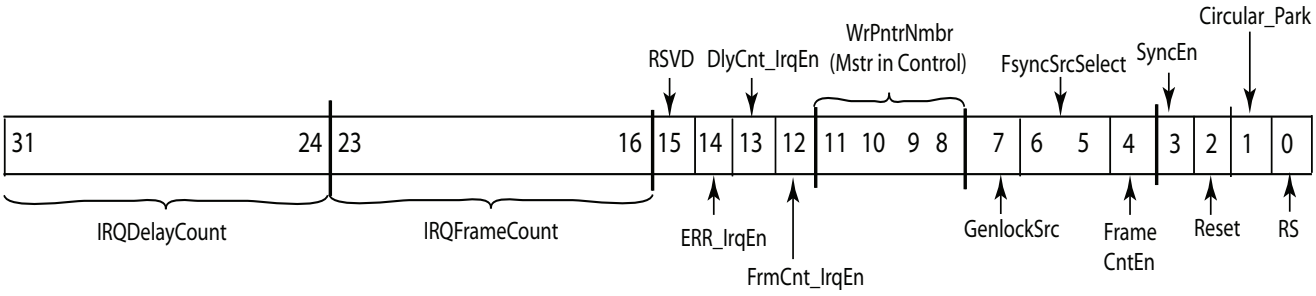


Figure 2-12: **S2MM DMACR Register**

Table 2-19: **S2MM_DMACR Register Details**

Bits	Field Name	Default Value	Access Type	Description
31 downto 24	IRQDelayCount	0x00	R/W	This value is used for setting the interrupt delay count value. The delay count interrupt is a mechanism for causing the DMA engine to generate an interrupt after the delay period has expired. Timer begins counting either upon receipt of frame sync (if C_USE_FSYNC=1,2,3) or completion of vsize lines (if C_USE_FSYNC=0). It resets with subsequent start of packet (s_axis_s2mm_tvalid) assertion. When a value different than the current IRQDelayCount is written to this field, the internal delay counter is reset to the new value. ***Note:*** Setting this value to zero disables the delay counter interrupt.
23 downto 16	IRQFrameCount	01h	R/W	Interrupt Frame Count. This value is used for setting the interrupt threshold. When a frame transfer interrupt events occur, an internal counter counts down from the Interrupt Frame Count setting. On the frame boundary after count reaches 1, an interrupt out is generated by the VDMA engine. When an IRQFrameCount value, different than what is currently set, is written to the IRQFrameCount field, the internal frame counter is reset to the new IRQFrameCount value. ***Note:*** The minimum setting for the threshold is 0x01. A write of 0x00 to this register sets the threshold to 0x01. ***Note:*** When DMACR.FrameCntEn = 1, this value determines the number of frame buffers to process.
15	Reserved	0	RO	Writing to this bit has no effect and it is always read as zeros.
14	Err_IrqEn	0	R/W	Interrupt on Error Interrupt Enable. When set to 1, allows DMASR.Err_Irq to generate an interrupt out. • 0 = Error Interrupt disabled • 1 = Error Interrupt enabled
13	DlyCnt_IrqEn	0	R/W	Delay Count Interrupt Enable. When set to 1, allows DMASR.DlyCnt_Irq to generate an interrupt out. • 0 = Delay Count Interrupt disabled • 1 = Delay Count Interrupt enabled
12	FrmCnt_IrqEn	0	R/W	Frame Count Complete Interrupt Enable. When set to 1, allows DMASR.FrmCnt_Irq to generate an interrupt out when IRQFrameCount value reaches zero. • 0 = Frame Count Interrupt disabled • 1 = Frame Count Interrupt enabled

Table 2-19: **S2MM_DMACR Register Details *(Cont'd)***

Bits	Field Name	Default Value	Access Type	Description
11 downto 8	WrPntrNum	zeroes	R/W	Indicates the master in control when S2MM channel is configured for Genlock slave/Dynamic Genlock Master/Dynamic Genlock Slave (C_S2MM_GENLOCK_MODE = 1,2,3). • 0000b = Controlling entity is Entity 1 • 0001b = Controller entity is Entity 2 • 0010b = Controller entity is Entity 3 and so on ***Note:*** Maximum valid WrPntrNum is C_S2MM_GENLOCK_NUM_MASTER - 1. Setting to a value greater than C_S2MM_GENLOCK_NUM_MASTER - 1 has undefined results.
7	GenlockSrc	0	R/W	Sets the Genlock source for Genlock slaves • 0 = External Genlock • 1 = Internal Genlock This bit has meaning only: • if both VDMA channels are enabled AND • if one VDMA channel is configured as Genlock Master then the other VDMA channel *must* be configured as Genlock Slave OR if one VDMA channel is configured as Dynamic Genlock Master then other VDMA channel *must* be configured as Dynamic Genlock Slave AND • if C_INCLUDE_INTERNAL_GENLOCK = 1 See C_S2MM_GENLOCK_MODE in Parameter Descriptions for more details on different Genlock modes.
6 downto 5	FsyncSrcSelect	00	R/W	Selects the frame sync source for the S2MM channel. The frame sync source is selected as follows: • 00 = s2mm_fsync • 01 = mm2s_fsync • 10 = s_axis_s2mm_tuser(0) --] When C_S2MM_ENABLE_SOF = 1 else reserved • 11 = Reserved ***Note:*** Frame Sync Source select is valid only if AXI VDMA is configured in fsync mode(C_USE_FSYNC=1,2,3). ***Note:*** FsyncSrcSelect option 10b (s_axis_s2mm_tuser(0)) when SOF is enabled, C_S2MM_ENABLE_SOF=1. Else the selection is reserved.
4	FrameCntEn	0	R/W	Configures VDMA to allow only IRQFrameCount number of transfers to occur. After the IRQFrameCount frames are completely transferred, the axi_vdma channel halts, DMACR.RS bit is deasserted and the DMASR.Halted asserts when the channel has completely halted.

Table 2-19: **S2MM_DMACR Register Details *(Cont'd)***

Bits	Field Name	Default Value	Access Type	Description
3	SyncEn	0	R/W RO	Enables Genlock or Dynamic Genlock Synchronization. • 0 = Genlock or Dynamic Genlock Synchronization disabled. Genlock input is ignored by S2MM. • 1 = Genlock or Dynamic Genlock Synchronization enabled. S2MM synchronized to Genlock frame input. ***Note:*** This value is only valid when the channel is configured as Genlock Slave or Dynamic Genlock Master or Dynamic Genlock Slave(C_S2MM_GENLOCK_MODE = 1 or 2 or 3). If configured for Genlock Master mode (C_S2MM_GENLOCK_MODE = 0), this bit is reserved and always reads as zero. See C_S2MM_GENLOCK_MODE in Parameter Descriptions for more details on different Genlock modes. ***Note:*** R/W when C_S2MM_GENLOCK_MODE = 1, Reserved and RO when C_S2MM_GENLOCK_MODE = 0.
2	Reset	0	R/W	Soft reset for resetting the AXI VDMA S2MM channel. Setting this bit to a 1 causes the AXI VDMA S2MM channel to be reset. Reset is accomplished gracefully. Pending commands/transfers are flushed or completed. AXI4-Stream reset output is asserted. Setting DMACR.Reset = 1 only resets the S2MM channel. After completion of a soft reset, all S2MM registers and bits are in the Reset State. • 0 = Reset NOT in progress - Normal operation • 1 = Reset in progress
1	Circular_Park	1	R/W	When set to 1, indicates Circular Buffer Mode and frame buffers are processed in a circular manner. When set to 0, indicates Park Mode and channel will park on the frame buffer referenced by PARK_PTR_REG.RdFrmPntrRef. • 0 = Park – Engine will park on the frame buffer referenced by PARK_PTR_REG.RdFrmPntrRef. • 1 = Tail Pointer Mode/Circular Buffer Mode. SG Descriptor is processed until the TAILDESC pointer is reached if SG engine is included and start address is cycled through in a circular manner. ***Note:*** Transitions to/from Park and Circular Buffer modes occur on frame sync boundaries. ***Note:*** For Scatter Gather Mode (C_INCLUDE_SG = 1), Park Mode must only be enabled when AXI VDMA is Idle (DMASR.Idle = 1) and NOT halted (DMASR.Halted = 0). Undefined results occur if enabled at any other time. ***Note:*** For non-Scatter Gather Mode (C_INCLUDE_SG = 0), Park Mode can be specified at any time.

Table 2-19: **S2MM_DMACR Register Details *(Cont'd)***

Bits	Field Name	Default Value	Access Type	Description
0	RS	0	R/W	Run / Stop. Controls running and stopping of the VDMA channel. For any DMA operations to commence the AXI VDMA engine must be running (DMACR.RS=1). • 0 = Stop – VDMA stops when current (if any) DMA operations are complete. The halted bit in the DMA Status Register asserts to 1 when the DMA engine is halted. This bit gets cleared by the AXI VDMA hardware when an error occurs. The CPU can also choose to clear this bit to stop DMA operations. • 1 = Run – Start DMA operations. The halted bit in the DMA Status Register deasserts to 0 when the DMA engine begins operations. ***Note:*** If Run/Stop is cleared, in-progress stream transfers might terminate early.
RO = Read Only. Writing has no effect. R/W = Read / Write.				

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

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

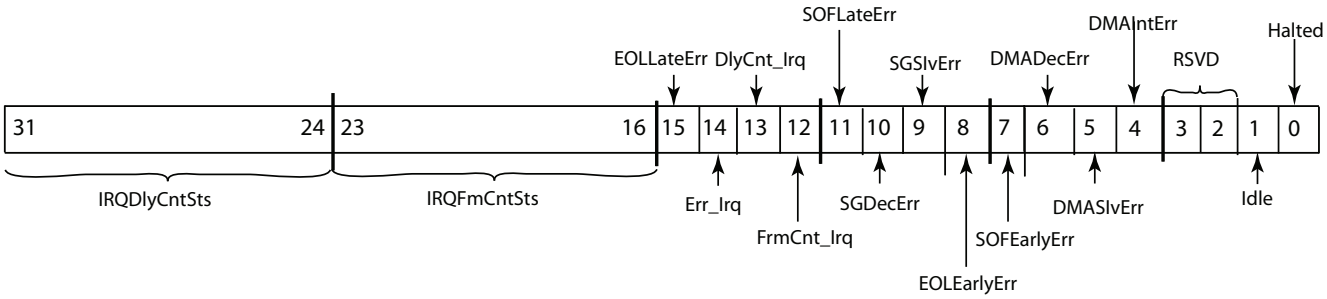


Figure 2-13: **S2MM DMASR Register**

Table 2-20: **S2MM_DMASR Register Details**

Bits	Field Name	Default Value	Access Type	Description
31 downto 24	IRQDelayCntSts	00h	RO	Interrupt Delay Count Status. Indicates current interrupt delay time value.
23 downto 16	IRQFrameCntSts	01h	RO	Interrupt Frame Count Status. Indicates current interrupt frame count value.

Table 2-20: **S2MM_DMASR Register Details *(Cont'd)***

Bits	Field Name	Default Value	Access Type	Description
15	EOLLateErr	0	R/WC	End of Line Late Error. • 0 = No End of Line Late Error • 1 = End of Line Late Error detected. VDMA does not halt This error occurs if incoming line size is greater than programmed hsize value. Write 1 to clear.
14	Err_Irq	0	R/WC	Interrupt on Error. When set to 1, indicates an interrupt event was generated on error. If enabled (DMACR.Err_IrqEn = 1), an interrupt out is generated from the AXI VDMA. 0 = No error Interrupt. 1 = Error interrupt detected.
13	DlyCnt_Irq	0	R/WC	Interrupt on Delay. Delay counter begins counting at the beginning of each frame and resets at delay count or transfer of video line. If delay count reached, the bit sets to 1, indicating an interrupt event was generated due to delay counter. If enabled (DMACR.DlyCnt_IrqEn = 1), an interrupt out is generated from the AXI VDMA. • 0 = No Delay Interrupt. • 1 = Delay Interrupt detected.
12	FrmCnt_Irq	0	R/WC	Frame Count Interrupt. When set to 1, indicates a Frame Count interrupt event was generated. This occurs when DMACR.FrameCount frames have been transferred. If enabled (DMACR.FrmCnt_IrqEn = 1) and if the interrupt threshold has been met, an interrupt out is generated from the AXI VDMA. • 0 = No Frame Count Interrupt. • 1 = Frame Count Interrupt detected.
11	SOFLateErr	0	R/WC	Start of Frame Late Error. • 0 = No Start of Frame Late Error • 1 = Start of Frame Late Error detected. VDMA does not halt This error occurs if incoming frame size is greater than programmed vsize value. Write 1 to Clear in flush on fsync mode.
10	SGDecErr	0	RO	Scatter Gather Decode Error. • 0 = No SG Decode Errors. • 1 = SG Decode Error detected. DMA Engine halts. See the Errors for more information.

Table 2-20: **S2MM_DMASR Register Details *(Cont'd)***

Bits	Field Name	Default Value	Access Type	Description
9	SGSlvErr	0	RO	Scatter Gather Slave Error. • 0 = No SG Slave Errors. • 1 = SG Slave Error detected. DMA Engine halts. See the Errors for more information.
8	EOLEarlyErr	0	R/WC	End of Line Early Error. • 0 = No End of Line Early Error • 1 = End of Line Early Error detected. VDMA does not halt This error occurs if incoming line size is lesser than programmed hsize value. Write 1 to clear.
7	SOFEarlyErr	0	RO or R/WC	Start of Frame Early Error. • 0 = No Start of Frame Early Error • 1 = Start of Frame Early Error detected. VDMA does not halt This error occurs if incoming frame size is lesser than programmed vsize value. Write 1 to Clear in flush on fsync mode and Read Only otherwise.
6	DMADecErr	0	RO	DMA Decode Error. • 0 = No DMA Decode Errors. • 1 = DMA Decode Error detected. DMA channel halts. This error occurs if the address request is to an invalid address.
5	DMASlvErr	0	RO	DMA Slave Error. • 0 = No DMA Slave Errors. • 1 = DMA Slave Error detected. DMA Engine halts. This error occurs if the slave read from the Memory Map interface issues a Slave Error.
4	DMAIntErr	0	RO or R/WC	DMA Internal Error. • 0 = No DMA Internal Errors. • 1 = DMA Internal Error detected. DMA channel halts. This error occurs during one of the following conditions: • Descriptor fetched with hsize or vsize = 0 in Scatter Gather mode • HSIZE or VSIZE register were written zeros in Register Direct mode • Received frame size is greater than vsize lines ***Note:*** In Flush On Frame Sync mode, this bit is R/WC (Write 1 to Clear) bit. Otherwise its a Read Only bit.

Table 2-20: **S2MM_DMASR Register Details *(Cont'd)***

Bits	Field Name	Default Value	Access Type	Description
3 downto 2	Reserved	0	RO	Writing to these bits has no effect and they are always read as zeros.
1	Idle	0	RO	DMA Scatter Gather Engine Idle. In Scatter Gather Mode (C_INCLUDE_SG = 1) this bit indicates the state of AXI VDMA Scatter Gather Engine operations. The assertion of Idle indicates the SG Engine has reached the tail pointer for the associated channel and all queued descriptors have been processed. If in the Idle state (DMASR.Idle = 1), writing to the TailPointer register automatically restarts DMA operations. For Register Direct Mode (C_INCLUDE_SG = 0) this bit is reserved and always read as 0b. • 0 = Not Idle — SG operations for S2MM channel in progress. • 1 = Idle — SG operation for S2MM channel paused. ***Note:*** DMASR.Idle only asserts after the SG engine has passed through the descriptor chain at least once and has reached the TAILDESC. ***Note:*** Writing to the TAILDESC register when not Idle (DMASR.Idle = 0) produces undefined results.
0	Halted	1	RO	DMA Channel Halted. Indicates the run/stop state of the DMA channel. • 0 = DMA channel running • 1 = DMA channel halted. This bit gets set when DMACR.RS = 0 and DMA and SG operations have halted. There can be a lag of time between when DMACR.RS = 0 and when DMASR.Halted = 1. ***Note:*** When halted (RS= 0 and Halted = 1), writing to CURDESC or TAILDESC pointer registers has no effect on DMA operations.
RO = Read Only. Writing has no effect R/WC = Read / Write to Clear. A CPU write of 1 clears the associated bit to 0.				

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

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

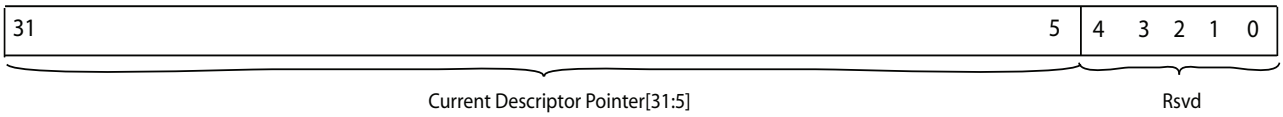


Figure 2-14: **S2MM CURDESC Register**

Table 2-21: **S2MM_CURDESC Register Details**

Bits	Field Name	Default Value	Access Type	Description
31 downto 5	Current Descriptor Pointer	zeros	R/W (RO)	In Scatter Gather Mode (C_INCLUDE_SG = 1) indicates the pointer of the current descriptor being worked on. This register must contain a pointer to a valid descriptor prior to writing to the TAILDESC register; otherwise, undefined results occur. When DMACR.RS is 1, CURDESC becomes Read Only (RO) and is used to fetch the first descriptor. When the DMA Engine is running (DMACR.RS=1), CURDESC registers are updated by AXI VDMA to indicate the current descriptor being worked on. On Scatter Gather error detection, CURDESC is updated to reflect the descriptor associated with the detected error. The register can only be written to by the CPU when the DMA Engine is Halted (DMACR.RS=0 and DMASR.Halted =1). At all other times, this register is Read Only (RO). Descriptors must be 8 word aligned, that is, 0x00, 0x20, 0x40, and so on. Any other alignment has undefined results. ***Note:*** In Register Direct Mode (C_INCLUDE_SG = 0) this field is reserved and always read as zeros.
4 downto 0 (Offset 0x38)	Reserved	0	RO	Writing to these bits has no effect, and they are always read as zeros.
RO = Read Only. Writing has no effect. R/W = Read / Write.				

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

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

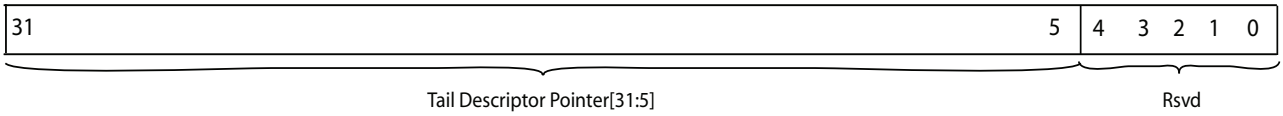


Figure 2-15: **S2MM TAILDESC Register**

Table 2-22: **S2MM_TAILDESC Register Details**

Bits	Field Name	Default Value	Access Type	Description
31 downto 5	Tail Descriptor Pointer	zeros	R/W (RO)	In Scatter Gather Mode (C_INCLUDE_SG = 1) indicates the pause pointer in a descriptor chain. The AXI VDMA SG Engine pauses descriptor fetching after completing operations on the descriptor whose current descriptor pointer matches the tail descriptor pointer. When AXI VDMA Channel is not Halted (DMASR.Halted = 0), a write by the CPU to the TAILDESC register causes the AXI VDMA SG Engine to start fetching descriptors or restart if it was idle (DMASR.Idle = 1). Writing to the TAILDESC when not idle (DMASR.Idle = 0) has undefined results. If the AXI DMA channel is halted (DMASR.Halted = 1 and DMACR.RS = 0), a write by the CPU to the TAILDESC register has no effect except to reposition the pause point. ***Note:*** Descriptors must be 8-word aligned, that is, 0x00, 0x20, 0x40, and so on. Any other alignment has undefined results. ***Note:*** In Register Direct Mode (C_INCLUDE_SG = 0) this field is reserved and always read as zeros.
4 downto 0	Reserved	0	RO	Writing to these bits has no effect, and they are always read as zeros.
RO = Read Only. Writing has no effect. R/W = Read / Write.				

S2MM_REG_INDEX (S2MM Register Index - Offset 44h) (C_INCLUDE_SG = 0)

This register provides access to upper bank of 16 (that is, 17 to 32) start addresses.

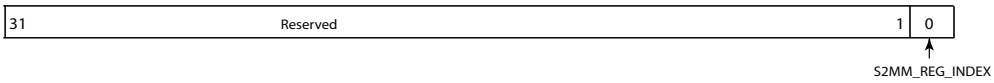


Figure 2-16: **S2MM Register Index**

Table 2-23: **S2MM Register Index (S2MM_REG_INDEX - Offset 0x44)**

Bits	Name	Default/Reset State	Access	Description
31 downto 1	Reserved		RO	Always read as zero
0	S2MM Reg Index	zeroes	RO	When set, enables access to the next set of 16 Frame Store Start Addresses (Bank1- 17 through 32) depending upon the following cases: Case 1: When C_NUM_FSTORES is less than or equal to 16, Bank1 (17 thru 32) registers are not available. Any writes to this bit do not change the behavior of VDMA. Case 2: When C_NUM_FSTORES is greater than 16 but less than 32 • 0 = Any write or read access between 0xAC to 0xE8 accesses the Bank0 (1 through 16) Frame StoreStart Address registers. • 1 = Accesses Bank1 registers. Example: If C_NUM_FSTORES = 20, Bank0 will have 1-16 Frame Store Start Addresses and Bank1 will have 17-20 Frame Store Start Addresses. Any access to Frame Store Start Addresses above 20 has no effect on writes and returns zero on reads. Case 3: When C_NUM_FSTORES is equal to 32 • 0 = Any write or read access between 0xAC to 0xE8 accesses the Bank0 registers. • 1 = Any write or read access between 0xAC to 0xE8 accesses the Bank1 registers.

Note: S2MM_REG_INDEX register is not present in case of SG =1 mode.

Note: The existing VDMA behavior of Dynamic S2MM Frame store selection (S2MM_FRMSTORE) remains unchanged with the addition of S2MM_REG_INDEX register.

S2MM_FRMSTORE (S2MM Frame Store Register – Offset 48h) (C_INCLUDE_SG = 1/0)

This register provides the number of Frame Stores to use for the Stream to Memory Map.

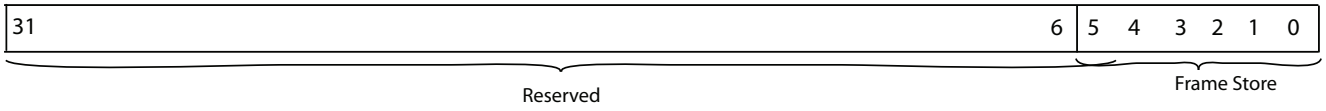


Figure 2-17: **S2MM_FRMSTORE Register**

Table 2-24: **S2MM_FRMSTORE Register Details**

Bits	Field Name	Default Value	Access Type	Description
31 downto 6	Reserved	0	RO	Writing to these bits has no effect, and they are always read as zeros.
5 downto 0	Frame Store	C_NUM_FSTORES	R/W	Indicates the number of frame stores to use for video data transfers. This value defaults to C_NUM_FSTORES. For Scatter Gather mode (C_INCLUDE_SG = 1) this value specifies the number of Scatter Gather Descriptors required. For Register Direct mode (C_INCLUDE_SG = 0) this value specifies the number of Start Address registers used for transfers. On reset and start-up this register is set to C_NUM_FSTORES. ***Note:*** Genlock Masters and their attached Genlock Slaves must have identical Frame Store settings. Any mismatch in values has undefined results. ***Note:*** Dynamic Genlock Masters and their attached Dynamic Genlock Slaves must have identical Frame Store settings. Any mismatch in values has undefined results. ***Note:*** Values written must be greater than 0 and less than or equal to C_NUM_FSTORES. Any other value has undefined results.
RO = Read Only R/W = Read / Write				

S2MM_THRESHOLD (MM2S Line Buffer Threshold Register – Offset 4Ch) (C_INCLUDE_SG = 1/0)

This register provides the Line Buffer Threshold for the Stream to Memory Map channel.

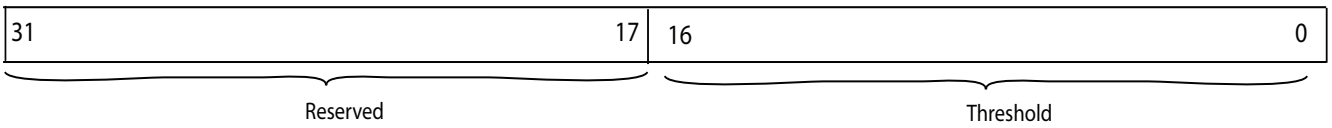


Figure 2-18: **S2MM_THRESHOLD Register**

Table 2-25: **S2MM_THRESHOLD Register Details**

Bits	Field Name	Default Value	Access Type	Description
31 downto 17	Reserved	zeroes	RO	Always read as zero
16 downto 0	Line Buffer Threshold	C_S2MM_ LINEBUFFER _THRESH	R/W	Threshold point at which S2MM line buffer almost full flag asserts high. Threshold specified in bytes and must be a multiple of C_S_AXIS_S2MM_TDATA_WIDTH/8, subject to the following condition. When Stream Data Width value is equal to a non-power of 2 value (that is, 24, 40, 72, 136, 264, 520), Threshold follows the restriction impose by the next nearest upper power of 2 value (that is, 32, 64, 128, 256, 512, 1024 respectively). Stream Data Width Allowed Values 8 1,2,3,..... 16 2,4,6,..... 24,32 4,8,12,..... 40 to 64 8,16,24,..... 72 to 128 16,32,48,..... 136 to 256 32,64,96,.... 264 to 512 64,128,192,... 520 to 1024 128,256,384,.. ***Note:*** Maximum threshold value limited by C_S2MM_LINEBUFFER_DEPTH. ***Note:*** Value valid when S2MM line buffer is included (C_S2MM_LINEBUFFER_DEPTH > 0).
RO = Read Only R/W = Read / Write				

MM2S Vertical Size (MM2S_VSIZE – Offset 0x50) (C_INCLUDE_SG = 0)

In Register Direct Mode (C_INCLUDE_SG = 0) the vertical size register has a dual purpose: first to hold the number of vertical lines, and second to be the mechanism for starting an MM2S transfer. If MM2S_DMACR.RS = 1, a write to this register transfers the video parameters and start addresses to an internal register block for DMA controller use. This register must be written last for a particular channel.

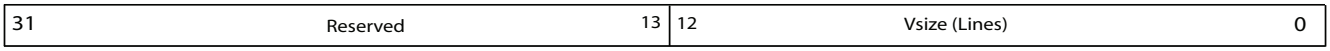


Figure 2-19: **MM2S VSIZE Register**

Table 2-26: **MM2S VSIZE Register Details**

Bits	Field Name	Default Value	Access Type	Description
31 downto 13	Reserved	zeros	RO	Writing to these bits has no effect, and they are always read as zeros.
12 downto 0	Vertical Size (Lines)	zeros	R/W	In Register Direct Mode (C_INCLUDE_SG = 0) indicates vertical size in lines of the video data to transfer. There are vsize number of packets that are hsize bytes long transmitted for each frame. In Scatter Gather Mode (C_INCLUDE_SG = 1) this field is reserved and always read as zero. ***Note:*** Writing to this register starts the VDMA transfers on MM2S Channel. Valid HSIZE, STRIDE, and Start Addresses must be set prior to writing MM2S_VSIZE or undefined results occur.
RO = Read Only. Writing has no effect. R/W = Read / Write.				

MM2S Horizontal Size (MM2S_HSIZE – Offset 0x54) (C_INCLUDE_SG = 0)

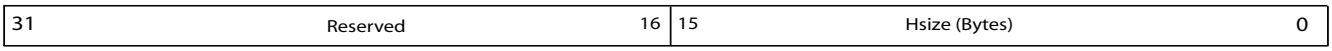


Figure 2-20: **MM2S HSIZE Register**

Table 2-27: **MM2S HSIZE Register Details**

Bits	Field Name	Default Value	Access Type	Description
31 downto 16	Reserved	zeros	RO	Writing to these bits has no effect, and they are always read as zeros.
15 downto 0	Horizontal Size (Bytes)	zeros	R/W	In Register Direct Mode (C_INCLUDE_SG = 0) indicates the horizontal size in bytes of the video data to transfer. There are vsize number of packets that are hsize bytes long transmitted for each frame. ***Note:*** A value of zero in this field when MM2S_VSIZE is written causes a DMAIntErr to be flagged in the DMASR Register.
RO = Read Only. Writing has no effect. R/W = Read / Write.				

MM2S Frame Delay and Stride (MM2S_FRMDLY_STRIDE – Offset 0x58) (C_INCLUDE_SG = 0)

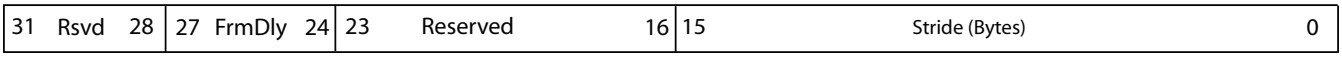


Figure 2-21: **MM2S Frame Delay and Stride Register**

Table 2-28: **MM2S FRMDELAY_STRIDE Register Details**

Bits	Field Name	Default Value	Access Type	Description
31 downto 29	Reserved	zeroes	RO	Writing to these bits has no effect, and they are always read as zeros.
28 downto 24	Frame Delay	zeros	R/W	In Register Direct Mode (C_INCLUDE_SG = 0) indicates the minimum number of frame stores the Genlock slave is to be behind the locked master. This field is only used if the channel is enabled for Genlock Slave operations (C_MM2S_GENLOCK_MODE = 1). This field has no meaning in other Genlock modes. In Scatter Gather Mode (C_INCLUDE_SG = 1) this field is reserved and always read as zero. ***Note:*** Frame Delay must be less than or equal to MM2S_FRMSTORE or undefined results occur.
23 downto 16	Reserved	zeros	RO	Writing to these bits has no effect, and they are always read as zeros.
15 downto 0	Stride (Bytes)	zeros	R/W	In Register Direct Mode (C_INCLUDE_SG = 0) indicates the number of address bytes between the first pixels of each video line. In Scatter Gather Mode (C_INCLUDE_SG = 1) this field is reserved and always read as zero. ***Note:*** A stride value less than MM2S_HSIZE causes data to be corrupted.
RO = Read Only. Writing has no effect. R/W = Read / Write.				

MM2S Start Addresses (Offsets 0x5C to Maximum Offset 0x98) (C_INCLUDE_SG = 0)

There are C_NUM_FSTORES start addresses for each channel. There is a maximum of 32 start registers available that are divided in two register banks, Bank0 and Bank1, each of 16 registers. Both the banks have the same initial offset (that is, 0x5C) and are accessed depending upon the MM2S_REG_INDEX value. If the user wants to access the 17th start address, it can be done by setting MM2S_REG_INDEX to 1 and accessing offset 0x5C.

31		0
31	Start Address 1	0
31	:	0
31	Start Address N(1)	0

1. N = C_NUM_FSTORES

Figure 2-22: **MM2S Start Address Register/s 1 to N**

Table 2-29: **MM2S Start Address Register Details**

Bits	Field Name	Default Value	Access Type	Description
31 downto 0 (Offset 0x5C)	Start Address 1	zeros	R/W	In Register Direct Mode (C_INCLUDE_SG = 0) indicates the Start Address for video buffer 1. This is the starting location for video data reads by MM2S. In Scatter Gather Mode (C_INCLUDE_SG = 1) this field is reserved and always read as zero.
31 downto 0 (Offset 0x60 to 0x98 max.)	Start Address 2 to Start Address N	zeros	R/W RO	In Register Direct Mode (C_INCLUDE_SG = 0) and Number of Frame Stores greater than 1 (C_NUM_FSTORES > 1) indicates the Start Addresses for video buffer 2 to video buffer N where N = C_NUM_FSTORES. In Scatter Gather Mode (C_INCLUDE_SG = 1) this field is reserved and always read as zero. ***Note:*** Start Address Registers greater than C_NUM_FSTORES are reserved and always read as zero. ***Note:*** MM2S_FRMSTORE specifies the number of Start Address registers that are processed.
N = C_NUM_FSTORES RO = Read Only. Writing has no effect. R/W = Read / Write.				

S2MM Vertical Size (MM2S_VSIZE – Offset 0xA0) (C_INCLUDE_SG = 0)

In Register Direct Mode (C_INCLUDE_SG = 0) the vertical size register has a dual purpose: first to hold the number of vertical lines, and second to be the mechanism for starting an S2MM transfer. If S2MM_DMACR.RS = 1, a write to this register transfers the video parameters and start addresses to an internal register block for DMA controller use. This register must be written last for a particular channel.

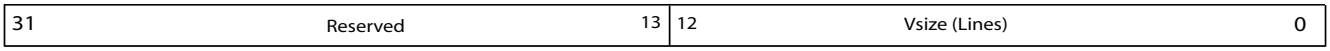


Figure 2-23: **S2MM VSIZE Register**

Table 2-30: **S2MM VSIZE Register Details**

Bits	Field Name	Default Value	Access Type	Description
31 downto 13	Reserved	zeros	RO	Writing to these bits has no effect, and they are always read as zeros.
12 downto 0	Vertical Size (Lines)	zeros	R/W	In Register Direct Mode (C_INCLUDE_SG = 0) indicates vertical size in lines of the video data to transfer. There are vsize number of packets that are hsize bytes long transmitted for each frame. In Scatter Gather Mode (C_INCLUDE_SG = 1) this field is reserved and always read as zero. ***Note:*** Writing to this register starts the VDMA transfers on S2MM channel. Valid HSIZE, STRIDE and Start Addresses must be set prior to writing S2MM_VSIZE or undefined results occur. ***Note:*** Writing a value of zero in this field causes a DMAIntErr to be flagged in the DMASR on next frame boundary.
RO = Read Only. Writing has no effect. R/W = Read / Write.				

S2MM Horizontal Size (S2MM_HSIZE – Offset 0xA4) (C_INCLUDE_SG = 0)

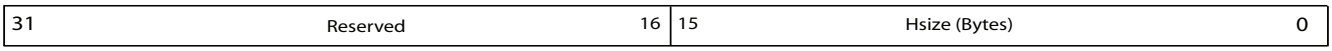


Figure 2-24: **S2MM HSIZE Register**

Table 2-31: **S2MM HSIZE Register Details**

Bits	Field Name	Default Value	Access Type	Description
31 downto 16	Reserved	zeros	RO	Writing to these bits has no effect, and they are always read as zeros.
15 downto 0	Horizontal Size (Bytes)	zeros	R/W	In Register Direct Mode (C_INCLUDE_SG = 0) indicates the horizontal size in bytes of the video data to transfer. The S2MM channel is configured to received vsize number of packets that are hsize bytes long for each frame. ***Note:*** A value of zero in this field when S2MM_VSIZE is written causes a DMAIntErr to be flagged in the DMASR Register on next frame boundary.
RO = Read Only. Writing has no effect. R/W = Read / Write.				

S2MM Frame Delay and Stride (S2MM_FRMDLY_STRIDE – Offset 0xA8) (C_INCLUDE_SG = 0)

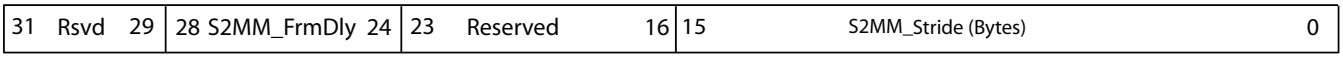


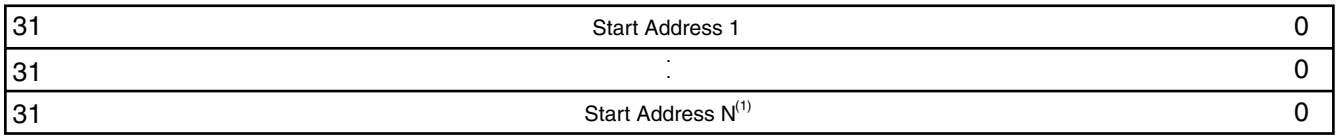
Figure 2-25: **S2MM Frame Delay and Stride Register**

Table 2-32: **S2MM FRMDELAY_STRIDE Register Details**

Bits	Field Name	Default Value	Access Type	Description
31 downto 29	Reserved	zeros	RO	Writing to these bits has no effect, and they are always read as zeros.
28 downto 24	Frame Delay	zeros	R/W	In Register Direct Mode (C_INCLUDE_SG = 0) indicates the minimum number of frame stores the Genlock slave is to be behind the locked master. This field is only used if channel is enabled for Genlock Slave Operations (C_S2MM_GENLOCK_MODE = 1) In Scatter Gather Mode (C_INCLUDE_SG = 1) this field is reserved and always read as zero. ***Note:*** Frame Delay must be less than or equal to S2MM_FRMSTORE or undefined results occur.
23 downto 16	Reserved	zeros	R/W	Writing to these bits has no effect, and they are always read as zeros.
15 downto 0	Stride (Bytes)	zeros	R/W	In Register Direct Mode (C_INCLUDE_SG = 0) indicates the number of address bytes between the first pixels of each video line. In Scatter Gather Mode (C_INCLUDE_SG = 1) this field is reserved and always read as zero. ***Note:*** A stride value less than S2MM_HSIZE causes data to be corrupted.
RO = Read Only. Writing has no effect. R/W = Read / Write.				

S2MM Start Addresses (Offsets 0xAC to Maximum Offset 0xE8) (C_INCLUDE_SG = 0)

There are C_NUM_FSTORES start addresses for each channel. There is a maximum of 32 start registers available that are divided in two register banks, Bank0 and Bank1, each of 16 registers. Both the banks have the same initial offset (that is, 0xE8) and are accessed depending upon the S2MM_REG_INDEX value. If the user wants to access the 17th start address, it can be done by setting S2MM_REG_INDEX to 1 and accessing offset 0xE8.



1. N = C_NUM_FSTORES

Figure 2-26: **S2MM Start Address Register/s 1 to N**

Table 2-33: **S2MM Start Address Register Details**

Bits	Field Name	Default Value	Access Type	Description
31 downto 0 (Offset 0xAC)	Start Address 1	zeros	R/W	In Register Direct Mode (C_INCLUDE_SG = 0) indicates the Start Address for video buffer 1. This is the starting location for video data writes by S2MM. In Scatter Gather Mode (C_INCLUDE_SG = 1) this field is reserved and always read as zero.
31 downto 0 (Offset 0xB0 to 0xE8 max.)	Start Address 2 to Start Address N	zeros	R/W RO	In Register Direct Mode (C_INCLUDE_SG = 0) and Number of Frame Stores greater than 1 (C_NUM_FSTORES > 1) indicates the Start Addresses for video buffer 2 to video buffer N where N = C_NUM_FSTORES. In Scatter Gather Mode (C_INCLUDE_SG = 1) this field is reserved and always read as zero. ***Note:*** Start Address Registers greater than C_NUM_FSTORES are reserved and always read as zero. ***Note:*** S2MM_FRMSTORE specifies the number of Start Address registers that are processed.
N = C_NUM_FSTORES RO = Read Only. Writing has no effect. R/W = Read / Write.				

S2MM hsize status Register S2MM_HSIZE_STATUS (offset 0xF0h) (C_INCLUDE_SG = 0/1)

This provides hsize count captured when first EOLEarlyErr occurs from incoming stream of S2MM channel. This register gets cleared automatically when S2MM_DMASR[8] is cleared.

Table 2-34: **S2MM hsize status Register Details**

Bits	Field Name	Default Value	Access Type	Description
31 downto 16	Reserved	0	RO	Writing to these bits has no effect, and they are always read as zeros.
15 downto 0	S2MMHsizeSts	0	RO	Indicates HSIZE count captured at first EOLEarlyErr error for S2MM channel.

S2MM vsize status Register S2MM_VSIZE_STATUS (offset 0xF4h) (C_INCLUDE_SG = 0/1)

This provides vsize count captured when first SOFEarlyErr occurs from incoming stream of S2MM channel. This register gets cleared automatically when S2MM_DMASR[7] is cleared.

Table 2-35: **S2MM vsize status Register Details**

Bits	Field Name	Default Value	Access Type	Description
31 downto 13	Reserved	0	RO	Writing to these bits has no effect, and they are always read as zeros.
12 downto 0	S2MMVsizeSts	0	RO	Indicates VSIZE count captured at first SOFEarlyErr error for S2MM channel.

Designing with the Core

General Design Guidelines

AXI VDMA is compliant to AXI4 on both the memory and streaming side. Any AXI compliant IP can be connected to the core. A typical MicroBlaze™ processor based system is shown in Figure 3-1. See XAPP741, XAPP742 for various system configuration using AXI VDMA.

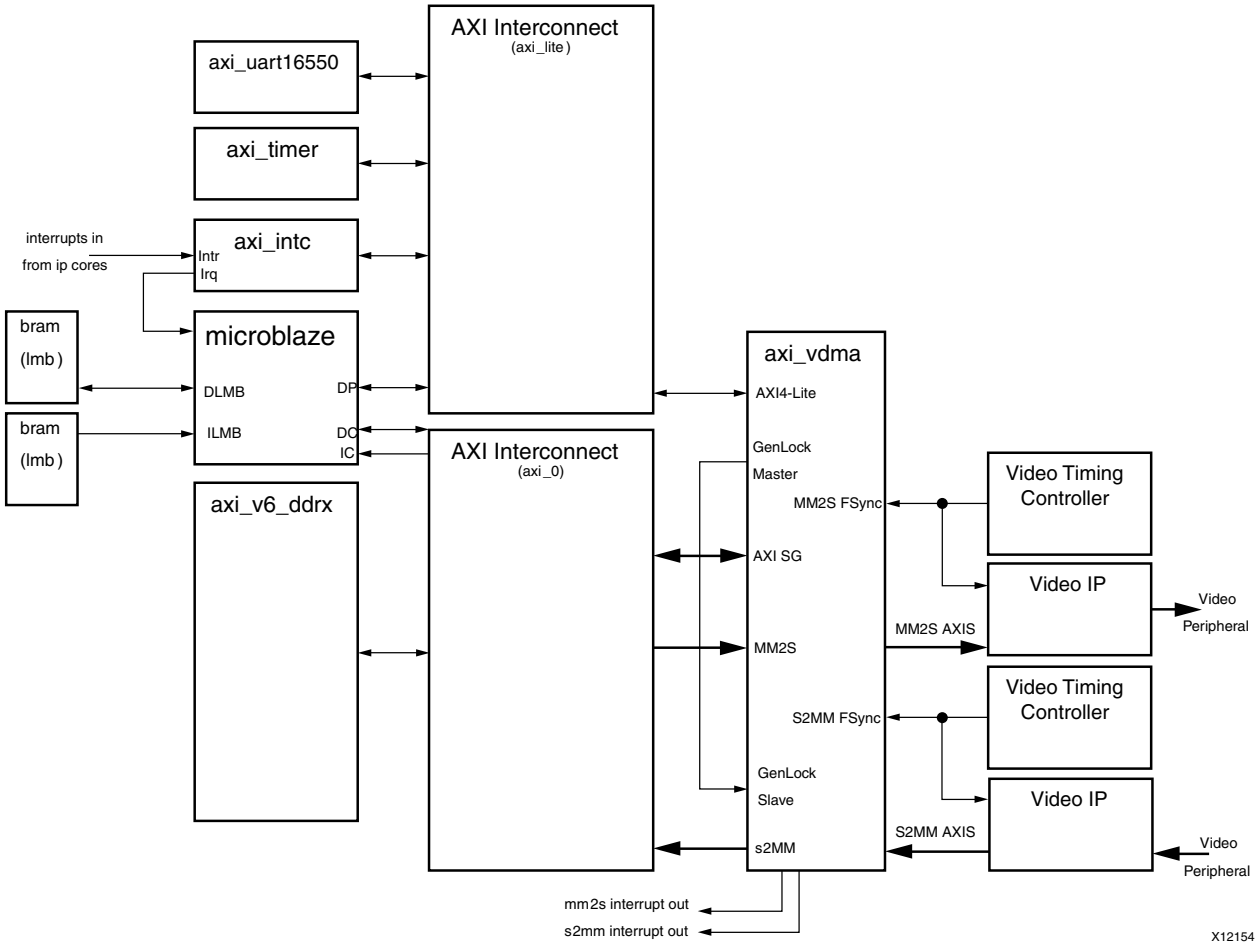


Figure 3-1: **Typical MicroBlaze Processor System Configuration**

Clocking

AXI VDMA provides two clocking modes of operation: asynchronous and synchronous. Asynchronous mode DMA control, Scatter Gather Engine, MM2S and S2MM Primary datapaths can all run asynchronously from each other. Setting C_PRMRY_IS_ACLK_ASYNC = 1 enables this mode and creates six clock domains.

- AXI4-Lite clock domain clocked by `s_axi_lite_aclk`
- Scatter Gather clock domain clocked by `m_axi_sg_aclk`
- mm2s clock domain on Memory Map side clocked by `m_axi_mm2s_aclk`
- s2mm clock domain on Memory Map side clocked by `m_axi_s2mm_aclk`
- s2mm clock domain on streaming side clocked by `s_axis_s2mm_aclk`
- mm2s clock domain on streaming side clocked by `m_axis_mm2s_aclk`

In synchronous mode, C_PRMRY_IS_ACLK_ASYNC = 0, all logic runs in a single clock domain. The signals `s_axi_lite_aclk`, `m_axi_sg_aclk`, `m_axi_mm2s_aclk`, `m_axi_s2mm_aclk`, `m_axis_mm2s_aclk`, and `s_axis_s2mm_aclk` must be tied to the same source otherwise undefined results occur.

Dynamic Clock Feature

Dynamic Clock changing is supported on the AXI4-Stream. This allows users to dynamically change the stream clocks (`m_axis_mm2s_aclk` and `s_axis_s2mm_clk`) without affecting the AXI Interconnect, AXI Memory controller, and so on.

Limitation

Changing the clock frequency during operation has a system-wide effect. Therefore, system considerations must be addressed to support dynamic clock changing. To fully support dynamic changing of all clocks, the AXI4 Memory Map targets must be able to support dynamic clock changes and the AXI4-Stream targets must be able to support dynamic clock changes. Support for dynamic clock changing on a system level is outside of AXI VDMA control and is beyond the scope of this document. As such dynamic clock changing (changing the clock frequency during run time) of `s_axi_lite_aclk`, `m_axi_sg_aclk`, `m_axi_mm2s_aclk`, and `m_axi_s2mm_aclk` is *not* supported. If during operation a clock change is required on `s_axi_lite_aclk`, `m_axi_sg_aclk`, `m_axi_mm2s_aclk`, or `m_axi_s2mm_aclk` then a system reset is required.

Dynamic Resolution

This feature allows you to dynamically change both the streaming clock and frame format without going through the reset cycle. For example, you will be able to switch between HD1080 to NTSC/PAL without going through reset cycle.

Limitation

The following sequence should be exercised before switching to another frame format

a. Write DMACR(0) = 0.
b. Poll for DMASR(0) to become 1.
c. Then Write DMACR(0) = 1.
d. Then Write new HSIZE & VSIZE.

Table 3-1 illustrates which signals and signal sets are clocked by what clock in asynchronous mode.

Table 3-1: **Asynchronous Mode Clock Distribution (C_PRMRY_IS_ACLK_ASYNC = 1)**

Target Clocks	C_INCLUDE_SG = 1	C_INCLUDE_SG = 0
s_axi_lite_aclk	All s_axi_lite_* Signals mm2s_introut s2mm_introut axi_resetn	All s_axi_lite_* Signals mm2s_introut s2mm_introut axi_resetn
m_axi_sg_aclk	All m_axi_sg_* Signals	NA
m_axi_mm2s_aclk	All m_axi_mm2s_* Signals mm2s_prmry_reset_out_n	All m_axi_mm2s_* Signals mm2s_prmry_reset_out_n
m_axi_s2mm_aclk	All m_axi_s2mm_* Signals s2mm_prmry_reset_out_n	All m_axi_s2mm_* Signals s2mm_prmry_reset_out_n
m_axis_mm2s_aclk	All m_axis_mm2s_* Signals mm2s_axis_resetn mm2s_fsync mm2s_fsync_out mm2s_prmtr_update mm2s_frame_ptr_in mm2s_frame_ptr_out mm2s_buffer_empty mm2s_buffer_almost_empty	All m_axis_mm2s_* Signals mm2s_axis_resetn mm2s_fsync mm2s_fsync_out mm2s_prmtr_update mm2s_frame_ptr_in mm2s_frame_ptr_out mm2s_buffer_empty mm2s_buffer_almost_empty
s_axis_s2mm_aclk	All s_axis_s2mm_* Signals s2mm_axis_resetn s2mm_fsync s2mm_fsync_out s2mm_prmtr_update s2mm_frame_ptr_in s2mm_frame_ptr_out s2mm_buffer_full s2mm_buffer_almost_full	All s_axis_s2mm_* Signals s2mm_axis_resetn s2mm_fsync s2mm_fsync_out s2mm_prmtr_update s2mm_frame_ptr_in s2mm_frame_ptr_out s2mm_buffer_full s2mm_buffer_almost_full

Resets

The AXI VDMA uses a single Active-Low reset input signal `axi_resetn`. When this signal is asserted Low, it resets the entire AXI VDMA core. The reset signal must be synchronous to the `s_axi_lite_aclk` signal and should be asserted for a minimum of eight clock cycles of the slowest clock. This is considered a hard reset and there is no graceful completion of transactions; all registers are reset to power-on conditions; all queues are flushed; all internal logic is returned to power-on conditions.

The reset input `axi_aresetn` is registered on input to break long timing paths and help with timing closure. Also in reset distribution, the resets out of the reset module are registered for improved timing closure. These pipeline delays along with clock domain crossing logic when in asynchronous mode cause increased reset response time in AXI VDMA and specifies minimum/maximum reset pulse times.

AXI VDMA also provides Soft Reset with the DMA Control Register for each channel. Issuing a Soft Reset by setting MM2S DMA Control Register's Reset bit to 1 or S2MM DMA Control Register's Reset bit to 1 causes the respective channel to reset gracefully. Ongoing transfers will complete including any queued transfers. Resetting one channel with the DMA Control Register does *not* Reset the other channel.

AXI VDMA also provides separate reset output signals MM2S Reset Out and S2MM Reset Out on both channels.

Parameter Descriptions

C_PRMRY_IS_ACLK_ASYNC

- **Type:** Integer
- **Allowed Values:** 0,1 (default = 0)
- **Definition:** 0 = `s_axi_lite_aclk`, `m_axi_sg_aclk`, `m_axi_mm2s_aclk`, `m_axis_mm2s_aclk`, `m_axi_s2mm_aclk` and `s_axis_s2mm_aclk` are synchronous to each other; 1 = `s_axi_lite_aclk`, `m_axi_sg_aclk`, `m_axi_mm2s_aclk`, `m_axis_mm2s_aclk`, `m_axi_s2mm_aclk` and `s_axis_s2mm_aclk` are asynchronous to each other.
- **Description:** Provides ability to operate the primary datapath asynchronously to the AXI4-Lite and Scatter Gather Engine. This is used for applications where there is a requirement to operate the primary datapath at high frequencies, but this same high frequency requirement is not required for reading and writing control registers or for fetching and updating descriptors. In some cases, this allows for easier placement and timing closure at system build time. The EDK tool suite assigns this parameter based on the clock sources for `s_axi_lite_aclk`, `m_axi_sg_aclk`, `m_axi_mm2s_aclk`, `m_axis_mm2s_aclk`, `m_axi_s2mm_aclk` and `s_axis_s2mm_aclk`.

C_DLYTMR_RESOLUTION

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

C_NUM_FSTORES

- **Type:** Integer
- **Allowed Values:** 1 to 32 (default = 3)
- **Definition:** Maximum number of frame stores
- **Description:** This integer parameter is used to define the maximum number of frame storage locations to be processed by the AXI VDMA. For Scatter Gather Mode (C_INCLUDE_SG = 1), this parameter also defines the maximum number of Scatter Gather descriptors per channel in the descriptor chain required to initialize the AXI VDMA. For Register Direct Mode (C_INCLUDE_SG = 0), this parameter defines the maximum number of video Start Address Registers for each channel. The actual number of frame store locations used, per channel, is set by register write to MM2S_FRM_STORE (offset 0x18) or S2MM_FRM_STORE (offset 0x48) register for the associated channel.

C_USE_FSYNC

- **Type:** Integer
- **Allowed Values:** 0, 1, 2, 3 (default = 0)
- **Definition:** 0 = Free run mode, 1 = Both channels in frame sync mode, 2 = Only MM2S channel in frame sync mode, 3 = Only S2MM channel in frame sync mode
- **Description:** This integer parameter is used to set the synchronization mode of AXI VDMA. When in free mode, the AXI VDMA transfers data as quickly as it is able to. When in frame sync mode, the AXI VDMA transfers data starting with the falling edge of each `mm2s_fsync` or `s2mm_fsync` for the associated channel.

C_ENABLE_VIDPRMTR_READS

- **Type:** Integer
- **Allowed Values:** 0, 1 (default =1)
- **Definition:** 0 = Disable Video Parameter Reads, 1 = Enable Video Parameter Reads
- **Description:** This integer parameter is used to enable the read access to the video parameters by the `s_axi_lite` interface when configured for Register Direct Mode. Disabling the video parameter register reads reduces FPGA resource utilization.

C_INCLUDE_INTERNAL_GENLOCK

- **Type**: Integer
- **Allowed Values:** 0,1, (default = 0)
- **Definition:** 0 = Exclude internal Genlock bus, 1 = Include internal Genlock bus
- **Description**: Include or exclude an internal Genlock bus. It allows internal routing of MM2S and S2MM Genlock buses without having it connected outside the core.

C_MM2S_SOF_ENABLE

- **Type**: Integer
- **Allowed Values**: 0,1, (default = 0)
- **Definition**: 0 = Disables SOF generation on `m_axis_mm2s_tuser(0)`, 1 = Enables SOF generation on `m_axis_mm2s_tuser(0)`
- **Description**: SOF pulse is driven on `m_axis_mm2s_tuser(0)` coincident with first pixel of the first line for each frame. For additional information, see the Video IP: AXI Feature Adoption section of the UG761 *AXI Reference Guide*.

C_S2MM_SOF_ENABLE

- **Type**: Integer
- **Allowed Values**: 0,1, (default = 0)
- **Definition**: 0 = Disables SOF detection on `s_axis_s2mm_tuser(0)`, 1 = Enables SOF detection on `s_axis_s2mm_tuser(0)`
- **Description**: When S2MM channel is in external fsync mode (C_USE_FSYNC = 1,3), this setting along with FsyncSrcSelect = 10, enables SOF pulse detection on `m_axis_mm2s_tuser(0)`. SOF pulse is coincident with the first pixel of the first line for each frame. For additional information, see the "Video IP: AXI Feature Adoption" section of the *AXI Reference Guide* (UG761).

C_FLUSH_ON_FSYNC

- **Type:** Integer
- **Allowed Values:** 0, 1, 2, 3 (default = 1)
- **Definition:** 0 = No flush/reset on frame sync on any channel, 1 = Flush/reset on frame sync on both channels, 2 = Flush/reset on frame sync on MM2S channel but not on S2MM channel, 3 = Flush/reset on frame sync on S2MM channel but not on MM2S channel
- **Description:** Specifies when VDMA channel transactions are flushed and channel states are reset on frame sync.

C_S_AXI_LITE_ADDR_WIDTH

- **Type:** Integer
- **Allowed Values:** 9 (default = 9)
- **Definition:** Address bus width of attached AXI on the AXI4-Lite interface
- **Description:** This integer parameter is used by the AXI4-Lite interface to size the AXI read and write address bus related components within the Lite interface. The EDK tool suite assigns this parameter a fixed value of 9.

C_S_AXI_LITE_DATA_WIDTH

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

C_INCLUDE_SG

- **Type:** Integer
- **Allowed Values:** 0,1 (default = 0)
- **Definition:** 0 = Exclude SG Engine; 1 = Include SG Engine
- **Description:** Include or exclude Scatter Gather Engine. Setting this parameter to 0 causes all output ports for the Scatter Gather engine to be tied to zero and all of the input ports to be left open. Excluding the Scatter Gather engine configures the AXI VDMA for Register Direct Mode.

C_M_AXI_SG_DATA_WIDTH

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

C_M_AXI_SG_ADDR_WIDTH

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

C_INCLUDE_MM2S

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

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

C_INCLUDE_S2MM

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

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

C_INCLUDE_MM2S_DRE

- **Type:** Integer
- **Allowed Values:** 0,1 (default = 0)
- **Definition:** 0 = Exclude MM2S Data Realignment Engine; 1 = Include MM2S Data Realignment Engine
- **Description:** Include or exclude MM2S Data Realignment Engine. For use cases where all transfers are C_M_AXI_MM2S_DATA_WIDTH aligned, this parameter can be set to 0 to exclude DRE-saving FPGA resources. Setting this parameter to 1 allows data realignment to the byte (8 bits) level on the primary memory map datapaths. For the MM2S channel, vertical size (vsize) number of video lines each horizontal size (hsize) bytes long and spaced stride bytes apart (stride is number of bytes between first pixel of each line) are read from memory.

  For the case where C_INCLUDE_MM2S_DRE = 1, data reads can start from any Start Address byte offset, be of any horizontal size and stride value and the read data are aligned such that the first byte read is the first valid byte out on the AXI4-Stream.

  For the case where C_IN CLUDE_MM2S_DRE = 0, the Start Address must be aligned to multiples of C_M_AXI_MM2S_DATA_WIDTH bytes. Also Horizontal Size and Stride must be specified in even multiples of C_M_AXI_MM2S_DATA_WIDTH bytes.

  For example, if C_M_AXI_MM2S_DATA_WIDTH = 32, data are aligned if the Start Address at word offsets (32-bit offset), that is, 0x0, 0x4, 0x8, 0xC, and so on. Horizontal Size is 0x4, 0x8, 0xC and so on. Stride is 0x4, 0x8, 0xC, and so on.

  If C_M_AXI_MM2S_DATA_WIDTH = 64, data are aligned if the Start Address is at double-word offsets (64-bit offsets), that is, 0x0, 0x8, 0x10, 0x18, and so on. Horizontal Size and Stride are at 0x4, 0x8, 0xC, and so on.

  Note: If DRE is disabled (C_INCLUDE_MM2S_DRE = 0), unaligned start addresses, hsizes, or strides, are not supported. Having an unaligned Start Address, HSize, and/or Stride results in undefined behavior.

  Note: DRE support is only available for AXI4-Stream data width setting of 64-bits and less.

C_INCLUDE_S2MM_DRE

- **Type:** Integer
- **Allowed Values:** 0,1 (default = 0)
- **Definition:** 0 = Exclude S2MM Data Realignment Engine, 1 = Include S2MM Data Realignment Engine
- **Description:** Include or exclude S2MM Data Realignment Engine. For use cases where all transfers are C_M_AXI_S2MM_DATA_WIDTH aligned, this parameter can be set to 0 to exclude DRE-saving FPGA resources. Setting this parameter to 1 allows data realignment to the byte (8 bits) level on the primary memory map datapaths. For the S2MM channel, the vertical size (vsize) number of video lines each horizontal size (hsize) bytes long and spaced stride bytes apart (stride is number of bytes between first pixel of each line) are written to memory.

  For the case where C_INCLUDE_S2MM_DRE = 1, data writes can start from any Start Address byte offset, be of any horizontal size and stride value and the write data are aligned such that first valid byte in on the AXI4-Stream is the byte written to the memory location specified by the Start Address, Hsize, and Stride.

  For the case where C_IN CLUDE_S2MM_DRE = 0, then the Start Address must be aligned to multiples of C_M_AXI_S2MM_DATA_WIDTH bytes. Also Horizontal Size and Stride must be specified in even multiples of C_M_AXI_S2MM_DATA_WIDTH bytes. For example, if C_M_AXI_S2MM_DATA_WIDTH = 32, data are aligned if the Start Address at word offsets (32-bit offset), that is 0x0, 0x4, 0x8, 0xC, and so on, Horizontal Size is 0x4, 0x8, 0xC and so on, Stride is 0x4, 0x8, 0xC, and so on.

  If C_M_AXI_S2MM_DATA_WIDTH = 64, data are aligned if the Start Address is at double-word offsets (64-bit offsets), that is, 0x0, 0x8, 0x10, 0x18, and so on; Horizontal Size and Stride are at 0x4, 0x8, 0xC, and so on.

  Note: If DRE is disabled (C_INCLUDE_S2MM_DRE = 0), unaligned start addresses, hsizes, or strides, are not supported. Having an unaligned Start Address, HSize, and/or Stride results in undefined behavior. DRE support is only available for AXI4-Stream data width setting of 64-bits and under.

C_INCLUDE_MM2S_SF

- **Type:** Integer
- **Allowed Values:** 0,1 (default = 1)
- **Definition:** 0 = Exclude MM2S Store-And-Forward; 1 = Include MM2S Store-And-Forward
- **Description:** Include or exclude MM2S Store-And-Forward buffer. When included, a Store-And-Forward buffer and manager are instantiated in AXI VDMA preventing the MM2S channel from requesting more read data than can be held in the Store-And-Forward buffer. This is for use cases where the target Video IP cannot accept all of the stream data transmitted by AXI VDMA MM2S channel. After the Store-And-Forward buffer is full, the AXI VDMA MM2S channel does not issue any more read requests preventing the AXI4 Slave from being tied up.

  If both the MM2S Line Buffer and the Store-And-Forward buffer are included then `m_axis_mm2s_tvalid` does not assert high until the set line buffer threshold, C_MM2S_LINEBUFFER_THRESH, is met.

  When excluded, no Store-And-Forward buffer is instantiated and as soon as data is read by AXI VDMA, `m_axis_mm2s_tvalid` asserts. Also read requests by AXI VDMA are made as quickly as possible.

  Note: If both the MM2S Line Buffer (C_MM2S_LINEBUFFER_DEPTH /= 0) and the Store-And-Forward Buffer are included (C_INCLUDE_MM2S_SF = 1), both features share a single buffer saving block RAM resources. The depth of the buffer is set to the maximum of C_MM2S_LINEBUFFER_DEPTH or the necessary depth as required by Store-And-Forward, which is the depth that is the next power of two greater than 6 x C_MM2S_MAX_BURST_LENGTH.

C_INCLUDE_S2MM_SF

- **Type:** Integer
- **Allowed Values:** 0,1 (default = 1)
- **Definition:** 0 = Exclude S2MM Store-And-Forward; 1 = Include S2MM Store-And-Forward
- **Description:** Include or exclude S2MM Store-And-Forward buffer. When included, a Store-And-Forward buffer and manager are instantiated in the AXI VDMA preventing the S2MM channel from issuing write requests until it has enough data to fulfill the complete requested write. This is for use cases where the source Video IP cannot deliver a contiguous stream of data to be received by the AXI VDMA S2MM channel. After the Store-And-Forward buffer has enough data to complete a write, the AXI VDMA S2MM channel issues a write request. If there is not enough data for the write to complete, then no request is made preventing the AXI4 Slave from being tied up. When excluded, no Store-And-Forward buffer is instantiated and write requests by AXI VDMA are made as quickly as possible.

Note: If both the S2MM Line Buffer (C_S2MM_LINEBUFFER_DEPTH /= 0) and the Store-And-Forward Buffer are included (C_INCLUDE_S2MM_SF = 1), both features share a single buffer, saving block RAM resources. The depth of the buffer is set to the maximum of C_S2MM_LINEBUFFER_DEPTH or the necessary depth as required by Store-And-Forward, which is the depth that is the next power of two greater than 6 x C_S2MM_MAX_BURST_LENGTH.

C_MM2S_GENLOCK_MODE

- **Type:** Integer
- **Allowed Values:** 0, 1, 2, 3 (default = 0)
- **Definition:** 0 = Genlock Master Mode, 1 = Genlock Slave Mode, 2 = Dynamic Genlock Master Mode, 3 = Dynamic Genlock Slave Mode
- **Description**: This integer parameter sets the MM2S Channel Genlock synchronization mode.
  - **Genlock Master**: Frames are not dropped or repeated. `mm2s_frm_ptr_out` outputs the current frame number that Master is being worked on by the MM2S channel. Master does not know the status of Genlocked Slave. There is a possibility of Master stepping on to Slave's buffer if Frame Delay is not managed properly.
  - **Genlock Slave**: Slave tries to catch up with the Genlock Master either by dropping or repeating frames. It samples Genlocked Master frame number on `mm2s_frm_ptr_in` and operates with Frame Delay behind the Master. Slave does not outputs its current frame number on `mm2s_frm_ptr_out`.
  - **Dynamic Genlock Master**: In this mode, Master dynamically skips the frame buffers that Slave is operating on. Master outputs previously accessed frame pointer on `mm2s_frm_ptr_out`. It also samples `mm2s_frm_ptr_in` that contains the frame number that Dynamic Genlocked Slave is operating on to avoid stepping over.

    Example: In case of three frame stores, Dynamic Genlock Master will rotate around 0,1,2,0,1,2 etc as long as it is not stepping on Slave's buffer. If it does detect that it is stepping on Slave's buffer, it skips that buffer and keeps on rotating. Thus if Slave's buffer is 1 for long time, then the Master will rotate between 0,2,0,2 and so on.

    If C_GENLOCK_REPEAT_EN = 1, then it retains the previous frame pointer output in case of error-ed frame.
  - **Dynamic Genlock Slave**: In this mode, Slave operates on the frame number that Dynamic Genlocked Master is outputting. Frame Delay is not valid in Dynamic Genlock modes. Slave also drives its current accessed frame number on `mm2s_frm_ptr_out`.

    Note: Dynamic Genlock Master should be Genlocked only with Dynamic Genlock Slave to have meaningful results.

C_S2MM_GENLOCK_MODE

- **Type:** Integer
- **Allowed Values:** 0, 1, 2, 3 (default = 0)
- **Definition:** 0 = Genlock Master Mode, 1 = Genlock Slave Mode, 2 = Dynamic Genlock Master Mode, 3 = Dynamic Genlock Slave Mode
- **Description**: This integer values sets the S2MM channel Genlock synchronization mode.
  - **Genlock Master**: Frames are not dropped or repeated. `s2mm_frm_ptr_out` outputs the current frame number that Master is being worked on by the S2MM channel. Master does not know the status of Genlocked Slave. There is a possibility of Master of stepping on to Slave's buffer if Frame Delay is not managed properly.
  - **Genlock Slave**: Slave tries to catch up with the Genlock Master either by dropping or repeating frames. It samples Genlocked Master frame number on `s2mm_frm_ptr_in` and operates with Frame Delay behind the Master. Slave does not outputs its current frame number on `s2mm_frm_ptr_out`.
  - **Dynamic Genlock Master**: In this mode, Master dynamically skips the frame buffers that Slave is operating on. Master outputs previously written frame pointer on `s2mm_frm_ptr_out`. It also samples `s2mm_frm_ptr_in` that contains the frame number that Dynamic Genlocked Slave is operating on to avoid stepping over.

    Example: In case of three frame stores, Dynamic Genlock Master would rotate around 0,1,2,0,1,2 etc as long as it was not stepping on Slave's buffer. If it does detect that it would step on Slave's buffer, it would skip that buffer and keep on rotating. Thus if Slave's buffer is 1 for long time, then the Master would rotate between 0, 2, 0, 2 and so on.

    If C_GENLOCK_REPEAT_EN = 1, then it retains the previous frame pointer output in case of errored frame.
  - **Dynamic Genlock Slave**: In this mode, Slave follows the frame number that Dynamic Genlocked Master is outputting. Frame Delay is not valid in Dynamic Genlock modes. Slave also drives its current accessed frame number on `s2mm_frm_ptr_out`.

    Note: Dynamic Genlock Master should be Genlocked only with Dynamic Genlock Slave to have meaningful results.

C_MM2S_GENLOCK_NUM_MASTERS

- **Type:** Integer
- **Allowed Values:** 1 to 16(default = 1)
- **Definition:** Number of masters to which the slave synchronizes operations.
- **Description**: This integer parameter specifies the following:
  - For Genlock Slave, it specifies the number of Masters to which this Slave can synchronize.
  - For Dynamic Genlock Master, it specifies the number of Dynamic Genlock Slave can synchronize.
  - For Dynamic Genlock Slave, it specifies the number of Dynamic Genlock Master can synchronize.
  - This parameter also specifies the vector width of the `mm2s_frm_ptr_in` port, where each master requires 5 bits on the `mm2s_frm_ptr_in` vector. Therefore, the width of the `mm2s_frm_ptr_in` port is 6*C_MM2S_GENLOCK_NUM_MASTERS.

  Note: This parameter is not valid in Genlock Master mode(C_MM2S_GENLOCK_MODE = 0)

C_S2MM_GENLOCK_NUM_MASTERS

- **Type:** Integer
- **Allowed Values:** 1 to 16 (default = 1)
- **Definition:** Number of masters to which the slave synchronizes operations.
- **Description**: This integer parameter specifies the following:
  - For Genlock Slave, it specifies the number of Masters to which this Slave can synchronize.
  - For Dynamic Genlock Master, it specifies the number of Dynamic Genlock Slave can synchronize.
  - For Dynamic Genlock Slave, it specifies the number of Dynamic Genlock Master can synchronize.
  - This parameter also specifies the vector width of the `mm2s_frm_ptr_in` port, where each master requires 5 bits on the `s2mm_frm_ptr_in` vector. Therefore, the width of the `mm2s_frm_ptr_in` port is 6*C_S2MM_GENLOCK_NUM_MASTERS.

  Note: This parameter is not valid in Genlock Master mode(C_S2MM_GENLOCK_MODE = 0)

C_MM2S_GENLOCK_REPEAT_EN

- **Type:** Integer
- **Allowed Values:** 0 to 1(default = 0)
- **Definition:** 0 = Frame is advanced, 1 = Frame is repeated
- **Description**: Specifies if an errored frame is repeated on the next frame sync or if the frame is advanced to the next frame

  Note: This parameter is only valid for Genlock Slave mode (C_MM2S_GENLOCK_MODE = 0,2) with C_USE_FSYNC=1,2 and C_FLUSH_ON_FSYNC=1,2.

C_S2MM_GENLOCK_REPEAT_EN

- **Type:** Integer
- **Allowed Values:** 0 to 1(default = 0)
- **Definition:** 0 = Frame is advanced, 1 = Frame is repeated
- **Description**: Specifies if an errored frame is repeated on the next frame sync or if the frame is advanced to the next frame.

  This parameter is only valid for Genlock Slave mode (C_S2MM_GENLOCK_MODE = 0,2) with C_USE_FSYNC=1,3 and C_FLUSH_ON_FSYNC=1,3.

C_S_AXI_MM2S_ADDR_WIDTH

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

C_M_AXI_MM2S_DATA_WIDTH

- **Type:** Integer
- **Allowed Values:** 32, 64, 128, 256, 512, 1024 (default = 32)
- **Definition:** Data bus width of AXI on the AXI MM2S Memory Map Read interface
- **Description:** This integer parameter is used by the MM2S interface to size the AXI read data bus related components within the MM2S Channel. The EDK tools ensure correct sizing of the AXI data width based on EDK system configuration.

C_M_AXIS_MM2S_TDATA_WIDTH

- **Type:** Integer
- **Allowed Values:** Supports multiples of 8 bit widths; Min = 8; Max = 1024 (default = 32)
- **Definition:** Data bus width of AXI on the AXI MM2S Master Stream interface
- **Description:** This integer parameter is used by the MM2S interface to size the AXI Master Stream data bus-related components within the MM2S Channel.

  Note: This parameter must be set less than or equal to C_M_AXI_MM2S_DATA_WIDTH.

C_M_AXIS_MM2S_TUSER_BITS

- Type: Integer
- Allowed Values: 1
- Definition: 1 = Bit width of TUSER on MM2S interface is 1.

C_M_AXI_S2MM_ADDR_WIDTH

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

C_M_AXI_S2MM_DATA_WIDTH

- **Type:** Integer
- **Allowed Values:** 32, 64, 128, 256, 512, 1024 (default = 32)
- **Definition:** Data bus width of AXI on the AXI S2MM Memory Map Write interface
- **Description:** This integer parameter is used by the S2MM interface to size the AXI write data bus-related components within the S2MM channel. The EDK tools ensure correct sizing of the AXI data width based on the EDK system configuration.

C_S_AXIS_S2MM_TDATA_WIDTH

- **Type:** Integer
- **Allowed Values:** Supports multiples of 8 bit widths; Min = 8; Max = 1024 (default = 32)
- **Definition:** Data bus width of AXI on the AXI S2MM Slave Stream interface
- **Description:** This integer parameter is used by the S2MM interface to size the AXI Slave Stream data bus related components within the S2MM Channel.

  Note: This parameter must be set less than or equal to C_M_AXI_S2MM_DATA_WIDTH.

C_S_AXIS_S2MM_TUSER_BITS

- Type: Integer
- Allowed Values: 1
- Definition: 1 = Bit width of TUSER on S2MM interface is 1.

C_MM2S_MAX_BURST_LENGTH

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

C_S2MM_MAX_BURST_LENGTH

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

C_MM2S_LINEBUFFER_DEPTH

- **Type:** Integer
- **Allowed Values:** 0, 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536 (default = 128)
- **Definition:** MM2S line buffer depth
- **Description:** Depth of Line Buffer FIFO. Depth specified in bytes. Depth parameter must be a power of 2 value, that is, 1, 2, 4, 8, 16,...1024, 2048, and so on. The valid minimum depth, excluding 0, equals C_M_AXIS_MM2S_TDATA_WIDTH/8; it must always be a power of 2 value. In case this division produces a non-power of 2 value, the allowed minimum depth is the nearest upper power of 2 value.

Stream Data Width	Allowed Values
8	1, 2, 4,.....65536
16	2, 4, 8,.....65536
24, 32	4, 8, 16,.....65536
40 to 64	8, 16, 32,.....65536
72 to 128	16, 32, 64,.....65536
136 to 256	32, 64, 128,.....65536
264 to 512	64, 128, 256,....65536
520 to 1024	128, 256, 512,...65536

Note: A value of zero will exclude the line buffer.

C_S2MM_LINEBUFFER_DEPTH

- **Type:** Integer
- **Allowed Values:** 0, 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536 (default = 128)
- **Definition:** S2MM line buffer depth
- **Description:** Depth of Line Buffer FIFO. Depth specified in bytes. Depth parameter must be a power of 2 value, that is, 1, 2, 4, 8, 16,...1024, 2048, and so on. The valid minimum depth, excluding 0, equals C_S_AXIS_S2MM_TDATA_WIDTH/8; it must always be a power of 2 value. In case this division produces a non-power of 2 value, the allowed minimum depth is the nearest upper power of 2 value.

Stream Data Width	Allowed Values
8	1, 2, 4,.....65536
16	2, 4, 8,.....65536
24, 32	4, 8, 16,.....65536
40 to 64	8, 16, 32,.....65536
72 to 128	16, 32, 64,.....65536
136 to 256	32, 64, 128,.....65536
264 to 512	64, 128, 256,....65536
520 to 1024	128, 256, 512,...65536

Note: A value of zero will exclude the line buffer.

C_MM2S_LINEBUFFER_THRESH

- **Type:** Integer
- **Allowed Values:** 1 to 65536 (default = 4)
- **Definition:** MM2S line buffer almost empty threshold in bytes
- **Description:** Almost Empty Threshold. Sets the default threshold point at which the MM2S line buffer almost empty flag asserts high. The threshold can be modified dynamically during run time by writing to the MM2S_THRESHOLD register. The threshold is specified in bytes and must be multiple of C_M_AXIS_MM2S_TDATA_WIDTH/8. If C_M_AXIS_MM2S_TDATA_WIDTH is a non power of 2, the line buffer threshold value should be calculated based on the nearest upper power of 2 value.

For example if C_M_AXIS_MM2S_TDATA_WIDTH = 24, the threshold values should be calculated based on nearest upper power of 2. That is, C_M_AXIS_MM2S_TDATA_WIDTH = 32.

The following table illustrates the stream data width and its corresponding allowed line buffer threshold values

Stream Data Width	Allowed Values
8	1, 2, 3,.....
16	2, 4, 6,.....
24, 32	4, 8, 12,.....
40 to 64	8, 16, 24,.....
72 to 128	16, 32, 48,.....
136 to 256	32, 64, 96,....
264 to 512	64, 128, 192,...
520 to 1024	128, 256, 384,..

On power up and reset, the MM2S_THRESHOLD Register (Offset 0x1C) is set to this value.

Note: The maximum threshold value is limited by C_MM2S_LINEBUFFER_DEPTH

C_S2MM_LINEBUFFER_THRESH

- **Type:** Integer
- **Allowed Values:** 1 to 65536 (default = 4)
- **Definition:** S2MM line buffer almost full threshold in bytes
- **Description:** Almost Full Threshold. Sets the default threshold point at which the MM2S line buffer almost full flag asserts high. Threshold can be modified dynamically during run time by writing to the S2MM_THRESHOLD register. The threshold is specified in bytes and must be multiple of C_S_AXIS_S2MM_TDATA_WIDTH/8. If C_S_AXIS_S2MM_TDATA_WIDTH is a non-power of 2, the line buffer threshold value should be calculated based on the nearest upper power of 2 value.

  For example, if C_S_AXIS_S2MM_TDATA_WIDTH = 24, the threshold values should be calculated based on the nearest upper power of 2, that is, C_S_AXIS_S2MM_TDATA_WIDTH = 32.

The following table illustrates the stream data width and its corresponding allowed line buffer threshold values

Stream Data Width	Allowed Values
8	1, 2, 3,.....
16	2, 4, 6,.....
24, 32	4, 8, 12,.....
40 to 64	8, 16, 24,.....
72 to 128	16, 32, 48,.....
136 to 256	32, 64, 96,....
264 to 512	64, 128, 192,...
520 to 1024	128, 256, 384,..

On power up and reset, the S2MM_THRESHOLD Register (Offset 0x4C) is set to this value.

Note: The maximum threshold value is limited by C_S2MM_LINEBUFFER_DEPTH.

Core Implementation

Functional Simulation

VHDL and Verilog source files for axi_vdma_v5_02_a are provided un-encrypted for use in behavioral simulation within a simulation environment. Neither a test bench nor test fixture is provided with the AXI VDMA core.

Synthesis

Synthesis of the AXI VDMA can be performed with XST.

When using ISE® Design Suite Project Navigator, the Library for Verilog Sources for AXI VDMA needs to be set to axi_vdma_v5_02_a. This is done from the ISE Design Suite Project Navigator GUI on the Design tab by right-clicking on **Synthesize - XST**. Then select **Process Properties**... from the pop-up window. At the bottom of the dialog box, set the Property display level to 'Advanced' by selecting it from the drop-down menu. Finally under the Synthesis Options category, find the Property Name 'Library for Verilog Sources' and enter a value of axi_vdma_v5_02_a.

Xilinx Tools

See IP Facts for a list of tested design tools.

Static Timing Analysis

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

Sequence of Operation

Register Direct Mode (C_INCLUDE_SG = 0)

VDMA operations in register direct mode begin with the set up of the video parameter and start address registers and the DMA control registers. The following lists minimum steps, in order, required to start AXI VDMA operations:

1. Write control information to channel's DMACR register (Offset 0x00 for MM2S and 0x30 for S2MM) to set interrupt enables if desired, frame count, delay count if desired, and set DMACR.RS=1 to start the AXI VDMA channel running. There can be a lag between when the CPU sets DMACR.RS=1 and when AXI VDMA sets DMASR.Halted = 0. The CPU can determine if the AXI VDMA is running when DMACR.RS = 1 and DMASR.Halted = 0.
2. Write valid video frame buffer start address to the channel's START_ADDRESS register 1 to N where N equals C_NUM_FSTORES (Offset 058 up to 0x98 for MM2S and 0xAC up to 0xE8 for S2MM). Set the REG_INDEX register as and if required.
3. Write a valid Frame Delay and Stride to the channel's FRMDLY_STRIDE register (Offset 0x58 for MM2S and 0xA8 for S2MM).
4. Write a valid Horizontal Size to the channel's HSIZE register (Offset 0x54 for MM2S and 0xA4 for S2MM).
5. Finally write a valid Vertical Size to the channel's VSIZE register (Offset 0x50 for MM2S and 0xA0 for S2MM). This starts the channel transferring video data.

Note: On the S2MM channel, new video line size and number of video lines need to change following the assertion of `s2mm_prmtr_update` or undefined results occur.

Updating Video Transfer Information

In Register Direct Mode (C_INCLUDE_SG = 0), the user should be able to update video parameter settings at any time while the engine is running by writing new video parameters and a video start address through the AXI4-Lite control interface. The newly written video transfer values take effect on the next frame boundary after the user writes the vertical size register for the respective channel. The channel's `pmrtr_update` output (`mm2s_prmtr_update` and `s2mm_prmtr_update`) for the respective channel asserts, indicating the new video parameters are being used by the AXI VDMA.

To update video parameters dynamically while AXI VDMA operations are ongoing, a similar process to the start steps is needed.

1. Write the Frame Delay, Stride, and Horizontal Size in any order for the associated channel.
2. Finally, write the Vertical Size. When VSize is written, the video register values are transferred to an internal register block. On the next frame boundary the DMA controller for the associated channel starts transfers using the newly updated values.

For applications where reading of the video parameters and start address registers is not needed, the FPGA resources can be reduced by excluding the read logic for the video specific registers (VSIZE, HSIZE, FRMDLY_STRIDE, and START ADDRESS/ES). This is accomplished by setting C_ENABLE_VIDPRMTR_READS = 0.

Scatter Gather Mode (C_INCLUDE_SG = 1)

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

VDMA operations began with the setup of the descriptor pointer registers and the DMA control registers. The following lists minimum steps, in order, required for AXI VDMA operations:

1. Write a valid pointer to the channel's CURDESC_PNTR register (Offset 0x08 for MM2S and 0x38 for S2MM).
2. Write control information to channel's DMACR register (Offset 0x00 for MM2S and 0x30 for S2MM) to set interrupt enables if desired, frame count, delay count if desired, and set DMACR.RS=1 to start the AXI VDMA channel running. There can be a lag between when the CPU sets DMACR.RS=1 and when AXI VDMA sets DMASR.Halted = 0. The CPU can determine if the AXI VDMA is running when DMACR.RS = 1 and DMASR.Halted = 0.
3. Write a valid pointer to the channel's TAILDESC_PNTR register (Offset 0x10 for MM2S and 0x40 for S2MM). This starts the channel fetching and processing descriptors.

4. DMA scatter gather operations continue until the descriptor at TAILDESC_PNTR is processed, and then the engine idles as indicated by DMASR.Idle = 1.

The descriptor chain is made up of a maximum of C_NUM_FSTORES descriptors per channel. The SG Engine fetches descriptors and updates an internal register set with the descriptor information. The video timing information (vsize, hsize, stride, and frame delay) from the first descriptor fetched is captured and stored.

This video timing information is used for all transfers in all frames for the specified channel. All other video timing information in the other descriptors is ignored by the AXI VDMA. Start Addresses from each descriptor are populated into C_NUM_FSTORES internal address registers.

When the SG Engine reaches the descriptor pointed to by TAILDESC, the SG Engine processes that descriptor, sets the internal register set to ping-pong on the next frame sync (Internally generated if C_USE_FSYNC = 0 or externally if C_USE_FSYNC = 1,2,3), and enters an Idle state. At the next frame sync, the internal register set will switch such that the DMA controller operates on the newly updated values.

This method for handling data updates internally allows the SG engine to operate on a register set without stepping on the register set that the DMA controller is using for data transfers. It also allows the SG Engine to only fetch descriptors when a change is made to the data set, making it unnecessary for the SG engine to fetch a descriptor with each frame. The DMA controller continues to transfer video data even though the SG engine has reached the tail pointer and paused. This allows uninterrupted video data transfers.

Updating Video Transfer Information

To update the video transfer information for a channel, the CPU writes a new start address, vsize, hsize, stride, and frame delay information into the descriptor following the TAILDESC position. Then the CPU writes a new TAILDESC pointing to this newly updated descriptor. The SG Engine automatically fetches the newly updated descriptor and updates the start address, vsize, hsize, stride, and frame delay to the internal SG labeled register block. On the next frame sync for the channel these newly updated values are transferred to the Video labeled register block for use by the channel's DMA controller. The outputs, `mm2s_prmtr_update` and `s2mm_prmtr_update`, for the respective channel asserts coincident with `mm2s_fsync` and `s2mm_fsync` respectively when the new parameters take effect for the channel.

Note: On the S2MM channel, new video line size and number of video lines need to change following the assertion of `s2mm_prmtr_update` or undefined results occur.

The CPU should only update the TAILDESC when the engine is Idle; otherwise, undefined results occur. Updating the TAILDESC register when the SG Engine is Idle ensures that video timing information is not ignored in the descriptor.

The AXI VDMA Scatter Gather engine has independent descriptor queues for fetch descriptors for the MM2S and S2MM channels. Independent register sets and DMA controllers for MM2S and S2MM channels are also provided.

AXI VDMA General Operations

AXI VDMA Frame Boundary

A frame boundary depends on the mode of operation. For external fsync mode the frame boundary is based on the channel's fsync input, `mm2s_fsync` or `s2mm_fsync`. AXI VDMA starts the transfer for a particular frame after driving out an `fsync_out` for the associated channel (`mm2s_fsync_out` or `s2mm_fsync_out`). There is some pipeline delay between the fsync input and when `fsync_out` is driven.

For free run mode (C_USE_FSYNC = 0) an internal fsync is generated for timing transfers. AXI VDMA transfers the commanded transfers as quickly as it is able to process and transfer the data. AXI VDMA will drive out an `fsync_out` for the associated channel, `mm2s_fsync_out` or `s2mm_fsync_out`, on each frame boundary.

In S2MM channel, at the end of each frame, when all expected data are received, AXI VDMA continues to assert s_axis_s2mm_tready to avoid hang condition when C_USE_FSYNC=1,3 and C_FLUSH_ON_FSYNC=1,3. It also sets appropriate error bits as mentioned in Table 3-11 and Table 3-12.

When new video parameters have been updated to AXI VDMA and the associated channel starts operating on the new video parameters and start addresses, AXI VDMA asserts `mm2s_prmtr_update` or `s2mm_prmtr_update`, coincident with `fsync_out` for the associated channel. This indicates to the attached video IP that new line length and/or line number is in effect.

Note: It creates a deadlock scenario in S2MM channel if Streaming Master waits for `s2mm_prmtr_update` to assert before driving `s_axis_s2mm_tvalid` when `TUSER(0)` is used as SOF. It is because AXI VDMA expects `TUSER(0)` to be asserted in order to drive `s2mm_prmtr_update`. Hence it is recommended for Streaming Master not to depend on `s2mm_prmtr_update` to start the transfer.

AXI VDMA Video Transfer

Data is transferred from System Memory to Stream or Stream to System Memory as defined by the Start Address for the frame currently being operated on and the vertical size, horizontal size, and stride. Figure 3-2 illustrates a typical video image stored in system memory. The smaller portion of the video image is to be transferred. Vertical Size lines of Horizontal Size number of bytes are transferred from or to system memory starting at each start address for each video frame.

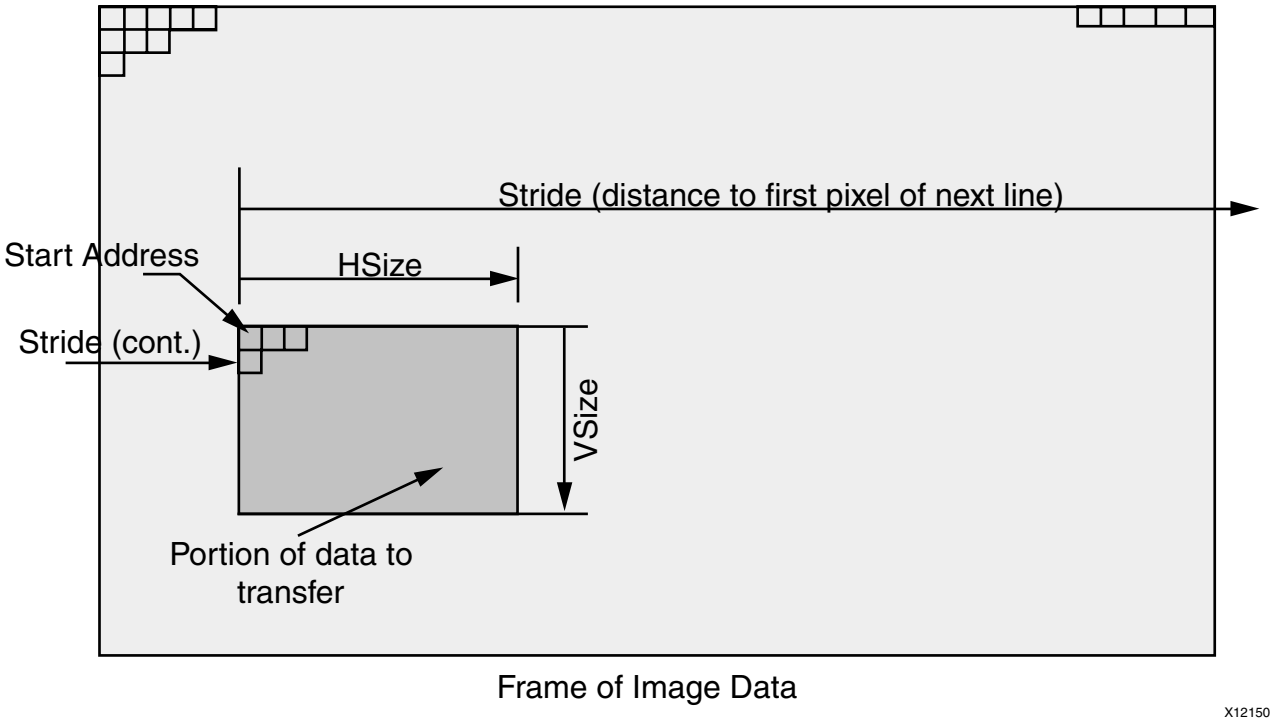


Figure 3-2: **Example Video Image Transfer**

Free Run Mode (C_USE_FSYNC = 0)

In free run mode (C_USE_FSYNC = 0), video data are transferred as quickly as possible. The output ports `mm2s_fsync_out` and `s2mm_fsync_out` indicate the frame boundaries. On start-up, that is, DMACR.RS set to 1 for the respective channel, an initial frame sync (`mm2s_fsync_out` or `s2mm_fsync_out`) asserts after the video parameters have been updated. This initial assertion of frame sync indicates the beginning of the first frame. For the MM2S channel, video data for a frame is transferred after the assertion of each `mm2s_fsync_out`. For the S2MM channel, axi_vdma expects new data for the frame to be transferred to the s2mm channel after the assertion of `s2mm_fsync_out`.

The AXI4-Stream output port, `s_axis_s2mm_tready`, asserts after `s2mm_fsync_out`, indicating the S2MM channel is ready to receive data. When all expected data (vsize lines that are hsize bytes long) have been received, AXI VDMA deasserts `s_axis_s2mm_tready` and does not reassert until the next `s2mm_fsync_out`. For both channels, MM2S and S2MM, when configured for free run mode, `mm2s_fsync_out` and `s2mm_fsync_out` asserts only after all of the data for a frame have been transferred. In free run mode the progression of video frames is in part controlled by the target Video IP on the MM2S or S2MM AXI4-Stream.

For MM2S, if m_axis_mm2s_tready is not asserted, then the frame time for mm2s is extended. Only after all video lines have been transferred is the frame considered complete and a new mm2s_fsync_out asserts. Likewise, on S2MM, if s_axis_s2mm_tvalid is not asserted, the frame time for s2mm is extended. Only after all video lines have been transferred is the frame considered complete and a new s2mm_fsync_out asserted.

Frame Sync Mode (C_USE_FSYNC = 1,2,3)

In frame sync mode, video data is synchronized to an external frame sync, mm2s_fsync for the MM2S channel and s2mm_fsync for the S2MM channel. The axi_vdma channel begins each frame on the falling edge of the associated channel's frame sync input, mm2s_fsync or s2mm_fsync. For the MM2S channel, data is driven out on the Master AXI4-Stream port following the assertion of mm2s_fsync. It is the responsibility of the target video IP to accept all transferred video lines before the assertion of the next mm2s_fsync or undefined results occur. For the S2MM channel, data is accepted on the Slave AXI4-Stream port following the assertion of s2mm_fsync. Again, the S2MM AXI4-Stream output port s_axis_s2mm_tready asserts after s2mm_fsync_out indicating that the S2MM channel is ready to receive data for a particular video frame.

It is the responsibility of the target video IP to transfer all video lines before the assertion of the next s2mm_fsync or undefined results occur.

The minimum time between assertions of the associated channel's fsync input, mm2s_fsync or s2mm_fsync, has been verified by simulation down to 1 µsec. Fsync rates faster than 1 µsec between assertions produce undefined results.

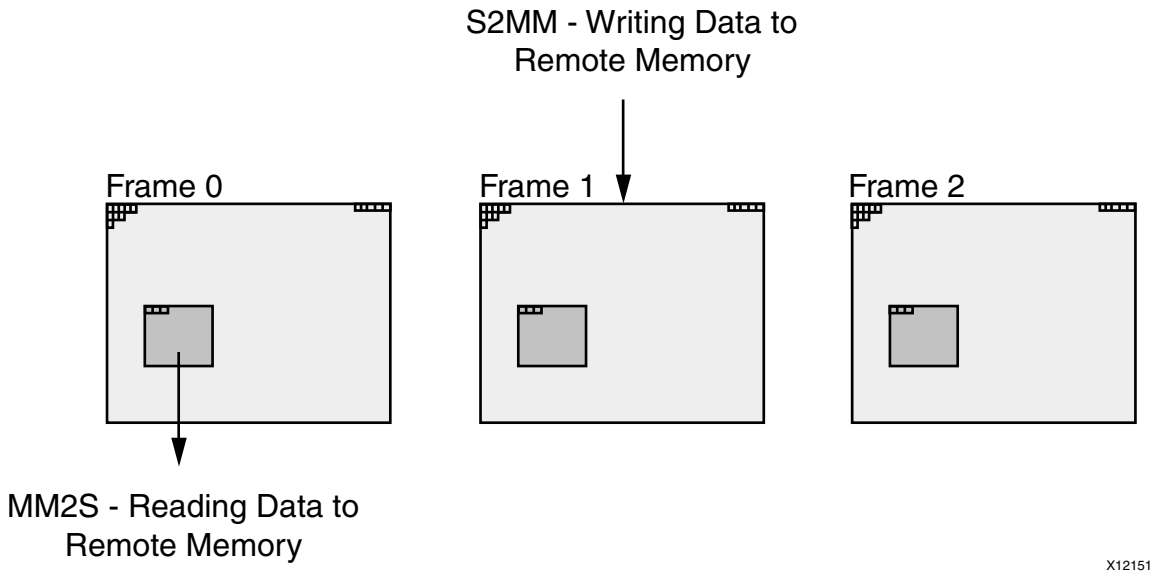


Figure 3-3: **Three Frame Example with S2MM and MM2S**

Genlock Synchronization

Figure 3-3 illustrates three frames of image transfer. MM2S channel reads from Frame 0 then on MM2S Frame Sync transitions to reading from Frame 1. Then Frame 2, and back to Frame 0 continuing this cycle with each Frame sync. S2MM writes to Frame 1 then transitions to Frame 2, followed by Frame 0. S2MM continues this progressing with each S2MM Frame Sync. To keep S2MM and MM2S from stepping on each other Genlock synchronization is used.

In many video applications, a producer of data runs at a different rate than the consumer of that data. To avoid the potential ill effects that such a rate mismatch can cause, frame buffering is often used. Frame buffering allocates multiple frames worth of memory to be used to hold the data. The data producer writes to one buffer while the consumer reads from another. See Figure 3-3. The two are kept in sync by not allowing both to use the same buffer at the same time. Typically one of the two is forced to either skip or repeat frames as necessary. This type of synchronization is called Genlock and is provided when the axi_vdma is configured for frame sync mode (C_USE_FSYNC = 1). Each channel of AXI VDMA has been designed to operate as either a Genlock Master or Slave. In general, a Genlock master specifies to a Genlock slave which frame to operate on at any given time.

Figure 3-4 illustrates simple timing of Genlock operation. In this example MM2S_FRMSTORE = 3, S2MM_FRMSTORE = 3, C_USE_FSYNC = 1, the frame delay for S2MM has been set to 1. Also, MM2S has been configured as the Genlock Master and S2MM channel has been configured as the Genlock Slave. The MM2S channel's frame rate is faster than that of S2MM so the S2MM Slave skips frames automatically. As one can see in Figure 3-4, in the time the MM2S channel cycles through frame 0, 1, 2 and back to 0, the S2MM channel has only cycled through two frame.

Due to the slow frame rate of S2MM compared to MM2S, the S2MM channel processes frame 2 then frame 0 then frame 2 again, skipping frame 1.

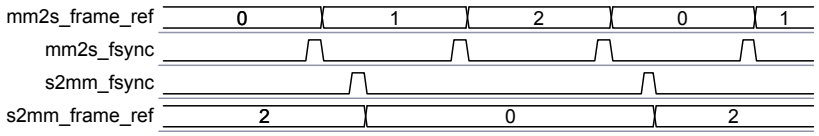


Figure 3-4: **Example Genlock Timing**

The Genlock mechanism that the AXI VDMA implements is based around the internal Start Address registers. The number of Start Address registers can be configured to be between 1 and 32. MM2S_FRMSTORE and S2MM_FRMSTORE specify this value for the associated channel with the C_NUM_FSTORES parameter specifying the maximum value. If large numbers of Frame Stores are not required, then specifying smaller values of C_NUM_FSTORE saves FPGA resources. The Genlock Master uses the index of the Start Address register to specify which Start Address register the Genlock Slave should use. This Start Address Register index is encoded as a Grey code value.

The Grey Code that is used depends upon the number of Frame Stores that was specified. Table 3-2 and Table 3-3 list the Grey Codes that are used for each of the 32 possible Frame Store sizes. The Grey Code cycles through all of the codes on the first line first and then cycles through all of the codes on the second line before repeating the first line. Number of grey codes is double the number of frame stores to allow for non-power-of-two frame store values to be cycled through and still maintain grey code coherency with minimal FPGA resources required.

Note: AXI VDMA does not support intra-frame mode. That is, Genlock synchronization cannot be provided for writing and reading the same frame with a predetermined delay between write and read operations.

Table 3-2: **Genlock Grey Code**

FRMSTORE	Grey Code
1	0 1
2	0 1 3 2
3	1 3 2 6 7 5
4	0 1 3 2 6 7 5 4
5	2 6 7 5 4 12 13 15 14 10
6	3 2 6 7 5 4 12 13 15 14 10 11
7	1 3 2 6 7 5 4 12 13 15 14 10 11 9
8	0 1 3 2 6 7 5 4 12 13 15 14 10 11 9 8
9	4 12 13 15 14 10 11 9 8 24 25 27 26 30 31 29 28 20
10	5 4 12 13 15 14 10 11 9 8 24 25 27 26 30 31 29 28 20 21
11	7 5 4 12 13 15 14 10 11 9 8 24 25 27 26 30 31 29 28 20 21 23
12	6 7 5 4 12 13 15 14 10 11 9 8 24 25 27 26 30 31 29 28 20 21 23 22
13	2 6 7 5 4 12 13 15 14 10 11 9 8 24 25 27 26 30 31 29 28 20 21 23 22 18
14	3 2 6 7 5 4 12 13 15 14 10 11 9 8 24 25 27 26 30 31 29 28 20 21 23 22 18 19
15	1 3 2 6 7 5 4 12 13 15 14 10 11 9 8 24 25 27 26 30 31 29 28 20 21 23 22 18 19 17
16	0 1 3 2 6 7 5 4 12 13 15 14 10 11 9 8 24 25 27 26 30 31 29 28 20 21 23 22 18 19 17 16

Table 3-3: **Gen-Lock Grey Codes (Frame Store 17 to 32)**

FrmStore (down) Frame number (right)	Grey Code															
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
17	8 48	24 49	25 51	27 50	26 54	30 55	31 53	29 52	28 60	20 61	21 63	23 62	22 58	18 59	19 57	17 56
18	9 48	8 49	24 51	25 50	27 54	26 55	30 53	31 52	29 60	28 61	20 63	21 62	23 58	22 59	18 57	19 56
19	11 48	9 49	8 51	24 50	25 54	27 55	26 53	30 52	31 60	29 61	28 63	20 62	21 58	23 59	22 57	18 56
20	10 48	11 49	9 51	8 50	24 54	25 55	27 53	26 52	30 60	31 61	29 63	28 62	20 58	21 59	23 57	22 56
21	14 48	10 49	11 51	9 50	8 54	24 55	25 53	27 52	26 60	30 61	31 63	29 62	28 58	20 59	21 57	23 56
22	15 48	14 49	10 51	11 50	9 54	8 55	24 53	25 52	27 60	26 61	30 63	31 62	29 58	28 59	20 57	21 56
23	13 48	15 49	14 51	10 50	11 54	9 55	8 53	24 52	25 60	27 61	26 63	30 62	31 58	29 59	28 57	20 56
24	12 48	13 49	15 51	14 50	10 54	11 55	9 53	8 52	24 60	25 61	27 63	26 62	30 58	31 59	29 57	28 56
25	4 48	12 49	13 51	15 50	14 54	10 55	11 53	9 52	8 60	24 61	25 63	27 62	26 58	30 59	31 57	29 56
26	5 48	4 49	12 51	13 50	15 54	14 55	10 53	11 52	9 60	8 61	24 63	25 62	27 58	26 59	30 57	31 56
27	7 48	5 49	4 51	12 50	13 54	15 55	14 53	10 52	11 60	9 61	8 63	24 62	25 58	27 59	26 57	30 56
28	6 48	7 49	5 51	4 50	12 54	13 55	15 53	14 52	10 60	11 61	9 63	8 62	24 58	25 59	27 57	26 56
29	2 48	6 49	7 51	5 50	4 54	12 55	13 53	15 52	14 60	10 61	11 63	9 62	8 58	24 59	25 57	27 56
30	3 48	2 49	6 51	7 50	5 54	4 55	12 53	13 52	15 60	14 61	10 63	11 62	9 58	8 59	24 57	25 56
31	1 48	3 49	2 51	6 50	7 54	5 55	4 53	12 52	13 60	15 61	14 63	10 62	11 58	9 59	8 57	24 56
32	0 48	1 49	3 51	2 50	6 54	7 55	5 53	4 52	12 60	13 61	15 63	14 62	10 58	11 59	9 57	8 56

The Grey codes received by the Genlock slave are then converted to a frame reference to tell the Genlock slave which frame to work on. The slave modifies the Genlock frame reference received by the frame delay such that the Genlock slaves remain a Frame Delay behind the Genlock Master. Table 3-4 illustrates an example conversion from Genlock Grey Code to Frame Reference used by the Genlock Slave.

Table 3-4: **Example Grey Code Conversion for NUMFSTORE = 5**

Grey To Frame Conversion	**Grey Code Progressions for NUMFSTORE = 5**									
Grey Code	2	6	7	5	4	12	13	15	14	10
Decoded Frame Reference	0	1	2	3	4	0	1	2	3	4

Additional Design Information

Scatter Gather Descriptor

When the AXI VDMA is configured for Scatter Gather Mode (C_INCLUDE_SG = 1), the Scatter Gather engine is used to pull in video transfer control information. This is accomplished by defining a linked list of transfer control information, referred to as a descriptor chain in system memory. A descriptor chain is required for each channel. The descriptor chain should be made up of MM2S_FRMSTORE and S2MM_FRMSTORE descriptors, respectively, where each descriptor is made up of seven 32-bit words. Each descriptor describes a video frame's worth of transfers. The following describes the descriptor fields. For Register Direct Mode (C_INCLUDE_SG = 0) the Scatter Gather engine is excluded and the video parameter and start address registers are accessed through the AXI4-Lite Control Interface.

IMPORTANT: *Descriptors must be aligned on eight 32-bit word alignment. Example valid offsets are 0x00, 0x20, 0x40, 0x60, and so on.*

Table 3-5: **Descriptor Fields**

Address Space Offset[a]	Name	Description
00h	NXTDESC	Next Descriptor Pointer points to the first word of the next descriptor to fetch.
04h	RESERVED	N/A
08h	START_ADDRESS	Start Address points to the starting pixel to transfer for the given frame.
0Ch	RESERVED	N/A
10h	VSIZE	Vertical Size specifies the number of video lines to transfer for the given frame.
14h	HSIZE	Horizontal Size specifies the size in bytes of the horizontal line to transfer for a given frame.
18h	FRMDLY_STRIDE	Stride specifies the number of bytes between the first pixels of each horizontal line. Frame Delay specifies the number of frames a Genlock slave should be behind the Genlock Master.

a. Address Space Offset is relative to eight 32-bit word alignment in system memory, that is, 0x00, 0x20, 0x40 and so on.

NXTDESC (Next Descriptor Pointer)

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

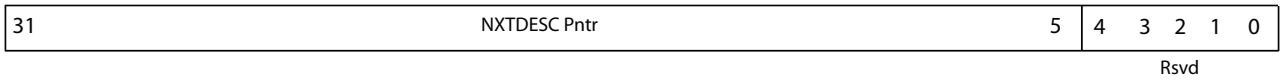


Figure 3-5: **NXTDESC**

Table 3-6: **NXTDESC Details**

Bits	Field Name	Description
31 downto 5	Next Descriptor Pointer	Indicates the lower order pointer pointing to the first word of the next descriptor. ***Note:*** Descriptors must be 8-word aligned, that is, 0x00, 0x20, 0x40, and so on. Any other alignment has undefined results.
4 downto 0	Reserved	These bits are reserved and should be set to zero.

START_ADDRESS (Start Address)

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

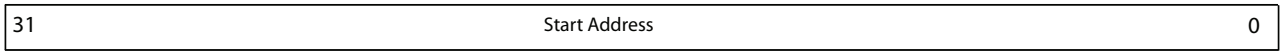


Figure 3-6: **Start Address**

Table 3-7: **START_ADDRESS Details**

Bits	Field Name	Description
31 downto 0	Start Address	Provides the starting pixel location of the data to transfer. For MM2S channel, data is read from system memory starting at this address. For S2MM channel, data is written to system memory starting at this address. ***Note:*** If the Data Realignment Engine is included (C_INCLUDE_MM2S_DRE = 1 or C_INCLUDE_S2MM_DRE = 1) for the respective channel, the Start Address can be at any byte offset. If the Data Realignment Engine is not included (C_INCLUDE_MM2S_DRE = 0 or C_INCLUDE_S2MM_DRE = 0) for the respective channel, the Start Address must be stream data width aligned.

VSIZE (Vertical Size)

This value provides vertical size (or number of lines) to transfer.

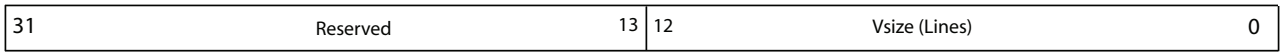


Figure 3-7: **VSIZE**

Table 3-8: **VSIZE Details**

Bits	Field Name	Description
31 downto 13	Reserved	This bit is reserved and should be written as zero.
12 downto 0	VSize	Vertical size in lines of the video data to transfer. On the MM2S stream interface there are vsize number of packets that are hsize bytes long transmitted for each frame. On the S2MM stream interface, the vsize number of packets that are hsize bytes long are expected to be received for each frame. ***Note:*** A value of zero on Vsize causes a MM2S_DMAIntErr or S2MM_DMAIntErr for the associated channel to be logged and the channel with the detected error is shut down.

HSIZE (Horizontal Size)

This value provides horizontal size in bytes of the video line to transfer.

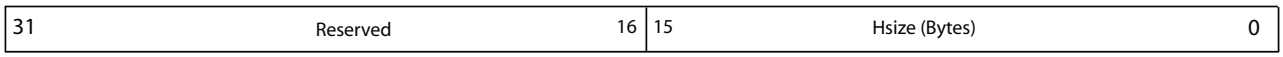


Figure 3-8: **HSIZE**

Table 3-9: **HSIZE Details**

Bits	Field Name	Description
31 downto 16	Reserved	These bits are reserved and should be set to zero.
15 downto 0	HSize	Horizontal size in bytes of the video data to transfer. On the MM2S stream interface there are vsize number of packets that are hsize bytes long transmitted for each frame. On the S2MM stream interface, vsize number of packets that are hsize bytes long are expected to be received for each frame. ***Note:*** If more or less data are received on the S2MM stream interface, an underrun or overrun error occurs. The channel will not halt on line size errors. ***Note:*** A value of zero for Hsize causes a MM2S_DMAIntErr or S2MM_DMAIntErr for the associated channel to be logged and the channel with the detected error is shut down.

FRMDLY_STRIDE (Frame Delay and Stride)

This value provides the Genlock Frame Delay and Stride for the video transfer.

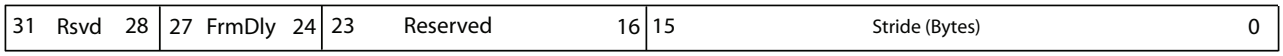


Figure 3-9: **FRMDLY_STRIDE**

Table 3-10: **FRMDLY_STRIDE Details**

Bits	Field Name	Description
31 downto 29	Reserved	These bits are reserved and should be set to zero.
28 downto 24	FrmDly	Frame Delay is the number of frame stores that the slave should be behind the locked master. This field is only used if the channel is enabled for Genlock Slave Operations (C_MM2S_GENLOCK_MODE = 1 or C_S2MM_GENLOCK_MODE=1) for the respective channel. ***Note:*** Frame Delay must be less than or equal to MM2S_FRMSTORE or S2MM_FRMSTORE for the respective channel or undefined results occur.
23 downto 16	Reserved	These bits are reserved and should be set to zero.
15 downto 0	Stride	Number of bytes between first pixels of each video line. ***Note:*** Stride values less than HSize results in corrupt data.

Line Buffers

An optional line buffer can be utilized to prevent memory controller throttling from causing inner packet throttling on the stream interface. The user can configure line buffer depth (with C_MM2S_LINEBUFFER_DEPTH and C_S2MM_LINEBUFFER_DEPTH) to provide the necessary buffering for a specific video application. For transmit (MM2S), this line buffering presents several clocks of latency at the onset of a video frame transfer between system memory read and initial tvalid on AXI4-Stream as the line buffer queues up a portion of the data to transfer.

Line buffer status is provided for video IP use. The signals `mm2s_linebuffer_empty` and `mm2s_linebuffer_almost_empty` are provided for the MM2S channel and `s2mm_linebuffer_full` and `s2mm_linebuffer_almost_full` are provided for the S2MM channel. Threshold parameters are provided to specify the assertion points for `mm2s_linebuffer_almost_empty` and `s2mm_linebuffer_almost_full`.

Note: Due to pipelining in the fifo_generated FIFOs and also clock crossing of asynchronous FIFOs, `s2mm_linebuffer_almost_full` and `mm2s_linebuffer_almost_empty` CANNOT be cycle accurate, meaning, for example, if the threshold is set to 8 bytes then `mm2s_linebuffer_almost_empty` does not deassert on the next clock cycle after the eighth byte is written into the FIFO. The `almost_empty` flag deasserts several (pipeline depth and possible clock crossing logic depth) cycles later.

Dynamic Line Buffer Threshold

For MM2S, the MM2S_THRESHOLD register (Offset 0x1C) specifies the assertion point for `mm2s_buffer_almost_empty`. The default value for this register is set by the C_MM2S_LINEBUFFER_THRESH parameter. The threshold register value is specified in bytes and must be a power of two value, that is, 1, 2, 4, 8, and so on. C_MM2S_LINEBUFFER_THRESH specifies the assertion point for `mm2s_buffer_almost_empty`. The `mm2s_buffer_almost_empty` signal asserts when there are MM2S_THRESHOLD bytes or less left in the Line Buffer. For example, if MM2S_THRESHOLD = 8 then `mm2s_buffer_almost_empty` asserts when the number of stored bytes is 8, 7, 6, 5, 4, 3, 2, or 1. When byte count = 0 then `mm2s_buffer_almost_empty` remains asserted and `mm2s_buffer_empty` will assert.

To prevent throttling within a MM2S AXI4-Stream packet, the target Video IP is responsible for not asserting `m_axis_mm2s_tready` = 1 until `mm2s_buffer_almost_empty` deasserts and `mm2s_buffer_empty` = 0, indicating more than MM2S_THRESHOLD data are stored in the line buffer. Adjustments might need to be made to the MM2S_THRESHOLD value for a user application to prevent throttling within packets.

For S2MM, the S2MM_THRESHOLD register (Offset 0x4C) specifies the assertion point for `s2mm_buffer_almost_full`. The default value for this register is set by the C_S2MM_LINEBUFFER_THRESH parameter. This allows for backwards compatibility with previous versions of AXI VDMA.

The threshold register value is specified in bytes and must be a power of 2 value, that is, 1, 2, 4, 8, and so on. S2MM_THRESHOLD specifies the assertion point of `s2mm_buffer_almost_full`. `s2mm_buffer_almost_full` asserts when there are S2MM_THRESHOLD bytes or more in the Line Buffer. For example, if S2MM_THRESHOLD = 8 then `s2mm_buffer_almost_full` asserts when the number of stored bytes is 8 or greater. When line buffer is full then both `s2mm_buffer_almost_full` and `s2mm_buffer_full` asserts. `s2mm_buffer_full` asserts when the last data beat of space is occupied. A data beat of space is C_S_AXIS_S2MM_TDATA_WIDTH bits wide.

Store-And-Forward

For cases where the target IP might throttle back on AXI VDMA, optional Store-And-Forward can be utilized. When included, a Store-And-Forward buffer and manager are instantiated in AXI VDMA. On MM2S, this prevents the channel from requesting more read data than can be held in the Store-And-Forward buffer. On S2MM this prevents the channel from issuing write requests when there is not enough data in the Store-And-Forward buffer to complete the write. For MM2S and S2MM, if the AXI4 data width equals the AXI4-Stream data width, that is, C_M_AXI_MM2S_DATA_WIDTH = C_M_AXIS_S2MM_TDATA_WIDTH and

C_M_AXI_S2MM_DATA_WIDTH = C_S_AXIS_S2MM_TDATA_WIDTH, and Store-And-Forward is included, AXI VDMA does not throttle back on the AXI4 Slave for reads or writes. For cases when the data widths do not match, the data has to be unpacked for MM2S and packed for S2MM, causing brief periodic throttles during the burst.

Additionally if Store-And-Forward is included and the Line Buffer for the associated channel is included, then both features share a common buffer the depth of which is determined by the maximum of the Line Buffer Depth setting and what is required for Store-And-Forward.

On MM2S, when Store-And-Forward and the Line Buffer are enabled, then the AXI4-Stream data valid (`m_axis_mm2s_tvalid`) does not assert until an initial threshold, C_MM2S_LINEBUFFER_THRESH, is met. This allows the target IP to ignore the side band `mm2s_buffer_almost_empty` flag and rely on the assertion of `m_axis_mm2s_tvalid` to determine when enough MM2S data has been queued.

Example MM2S Timing

Figure 3-10 illustrates example timing on MM2S channel for C_USE_FSYNC = 1, Vertical Size = 5 lines, Horizontal Size = 16, bytes, and Stride = 32 bytes. The figure shows the `m_axi_mm2s` and `m_axis_mm2s` interfaces.

Dataflow: After the reception of `mm2s_fsync`, AXI VDMA asserts `m_axi_mm2s_arvalid` with the start address on `m_axi_mm2s_araddr`. The signal `m_axi_mm2s_arvalid` is asserted five times to fetch five (vsize) lines of a frame. Read data from the mm side is stored in the line buffer and delivered on the streaming side by asserting `m_axis_mm2s_tvalid`. The signal `m_axis_mm2s_tlast` is asserted at the end of each line.

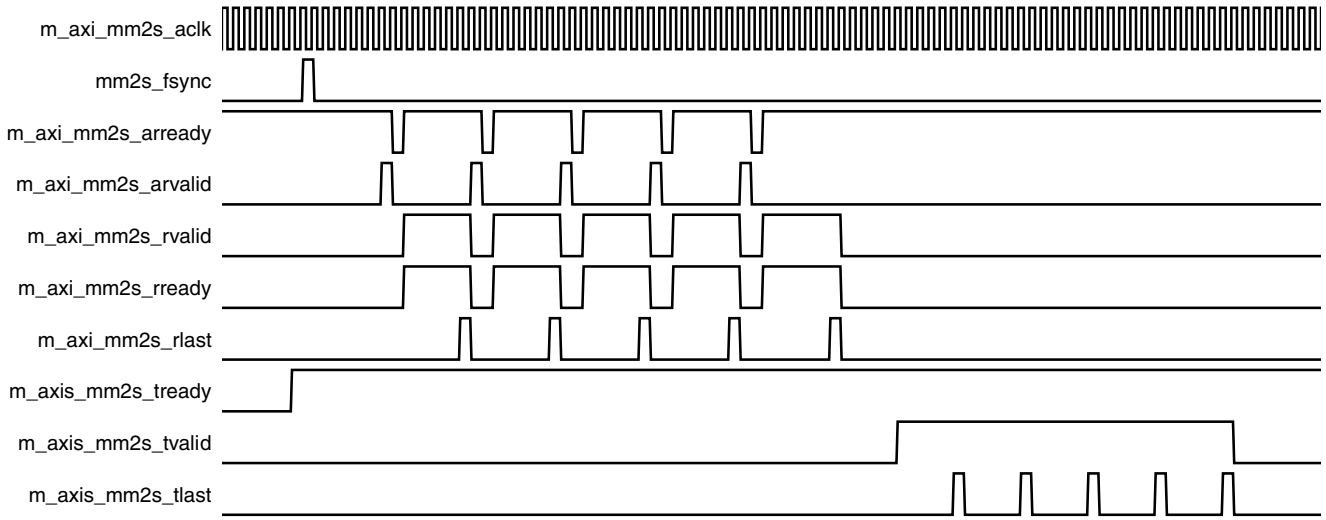


Figure 3-10: **Example MM2S Interface Timing**

Example S2MM Timing

Figure 3-11 illustrates example timing on S2MM channel for C_USE_FSYNC = 1, Vertical Size = 5 lines, Horizontal Size = 16, bytes, and Stride = 32 bytes. The figure shows the `m_axi_s2mm` and `s_axis_s2mm` interfaces.

Dataflow: After the reception of `s2mm_fsync`, AXI VDMA drives `s2mm_fsync_out` and `s_axis_s2mm_tready` to indicate its readiness to receive a frame on the streaming interface. Incoming streaming data is stored in the line buffer and driven onto the mm side by asserting `m_axi_s2mm_awvalid` and subsequently driving data on `m_axi_s2mm_wdata` along with `m_axi_s2mm_wvalid`.

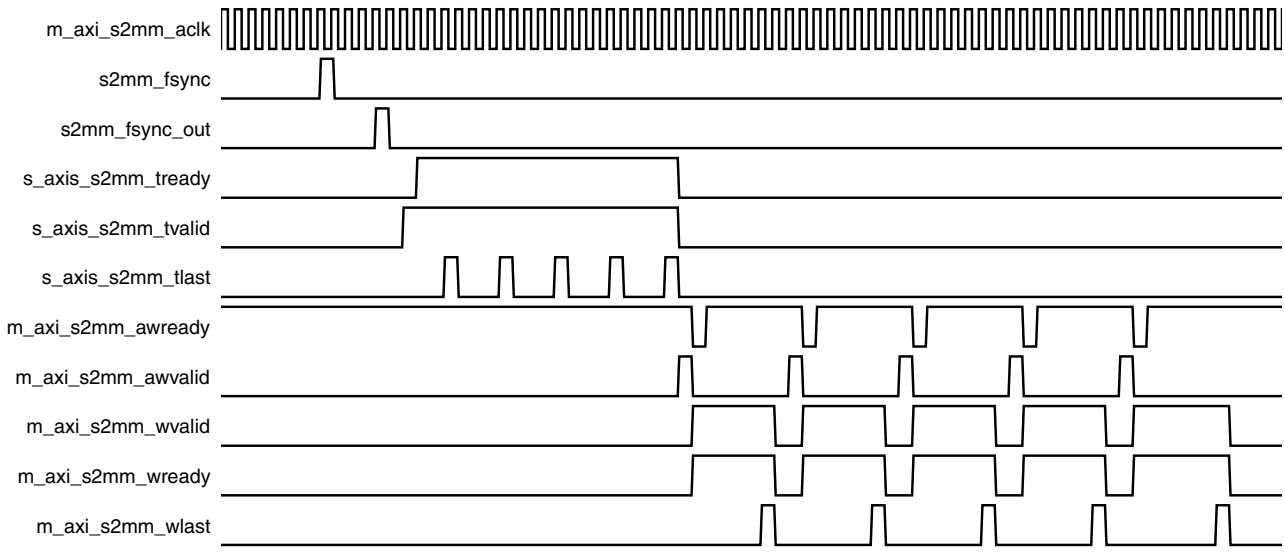


Figure 3-11: **Example S2MM Interface Timing**

Example Scatter Gather Timing

Figure 3-12 illustrates example timing for the Scatter Gather interface. The figure shows m_axi_sg interface timing with respect to m_axi_mm2s interface.

Dataflow: The SG engine starts when TAILDESC POINTER is written with a new value. C_NUM_FSTORES is programmed to 3. A chain of three descriptors are fetched. The signal mm2s_fsync is not sampled until AXI VDMA has completed fetching C_NUM_FSTORES of descriptors. On completion of descriptor fetching, AXI VDMA drives mm2s_fsync_out to signal a new frame start boundary. AXI VDMA then asserts m_axi_mm2s_arvalid with the start address on m_axi_mm2s_araddr. The signal m_axi_mm2s_arvalid is asserted five times to fetch five (vsize) lines of a frame. Read data from the mm side is stored in the line buffer and delivered on the streaming side by asserting m_axis_mm2s_tvalid. The signal m_axis_mm2s_tlast is asserted at the end of each line.

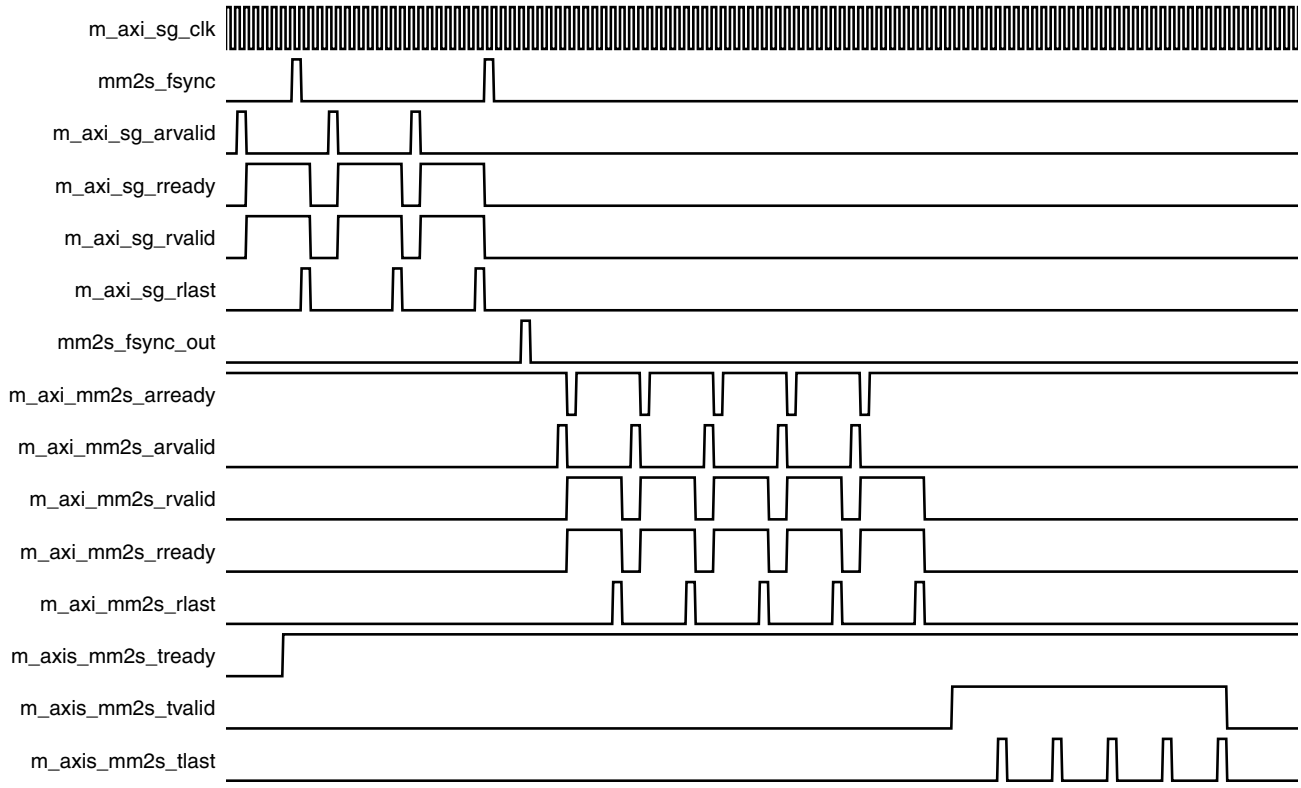


Figure 3-12: **Example Scatter Gather Timing**

AXI VDMA System Configuration

System configuration should be taken into consideration when using AXI VDMA. Due to the high performance and low latency design of AXI VDMA throttling or back pressure on the AXI4-Stream ports of AXI VDMA (MM2S and S2MM), this subsequently applies back pressure on the associated channel's AXI4 Memory Map side.

Depending on a user's system configuration, this back pressure can lead to a deadlock situation where, for example, a write transfer on S2MM to a single ported memory controller cannot complete because of a throttled read transfer on MM2S. A loopback type system where MM2S stream interface is looped back to S2MM stream interface can present such a deadlock scenario.

Two methods for resolving this deadlock scenario are provided, enabling the AXI Interconnect's FIFO blocks or including the Store-And-Forward feature on AXI VDMA.

AXI Interconnect Solution

The AXI Interconnect interfacing to the AXI VDMA AXI4 Memory Map ports on MM2S and S2MM can be configured to prevent the deadlock scenario described in the preceding paragraph. Read and Write Data FIFOs can be turned on in the AXI Interconnect to allow read and/or write data to be buffered up.

Enabling the FIFOs along with limiting the number of outstanding read and write requests accepted by AXI Interconnect guarantees that all requested data to be transferred can be accepted by the AXI Interconnect preventing deadlock.

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

- C_INTERCONNECT_M_AXI_MM2S_READ_ISSUING
- C_INTERCONNECT_M_AXI_MM2S_READ_FIFO_DEPTH
- C_INTERCONNECT_M_AXI_S2MM_WRITE_ISSUING
- C_INTERCONNECT_M_AXI_S2MM_WRITE_FIFO_DEPTH

Setting these parameters correctly configures the AXI Interconnect interfaced to the Memory Map ports of the AXI VDMA. For AXI VDMA, *_ISSUING multiplied by the associated channels *_MAX_BURST_LENGTH must be less than the *_FIFO_DEPTH to prevent a the deadlock scenario. For example, if C_MM2S_MAX_BURST_LENGTH is set to 16 and C_INTERCONNECT_M_AXI_MM2S_READ_ISSUING is set to 4, then the product of these two values is 16 X 4 = 64. Therefore, the setting C_INTERCONNECT_M_AXI_MM2S_READ_FIFO_DEPTH = 512 is sufficient to satisfy the requirements (16 X 4 = 64 which is less than 512). Following is the formula presented for clarity:

*_MAX_BURST_LENGTH X *_ISSUING < *_FIFO_DEPTH

When looking at FPGA resource utilization in an EDK system with AXI VDMA, note that the AXI Interconnect instantiates FIFOs for both MM2S and S2MM channels of AXI VDMA.

Store-And-Forward Solution

With this latest revision of AXI VDMA, an optional store and forward feature has been added. This is the recommended solution to the deadlock scenario described previously. To enable this feature, set C_INCLUDE_MM2S_SF = 1 and C_INCLUDE_S2MM_SF = 1.

Interrupt Controller

An interrupt output is provided for each channel (MM2S and S2MM). This output drives High when the interrupt frame count is met, if there is a delay interrupt, or an error if the associated interrupt is enabled, as shown in Figure 3-13.

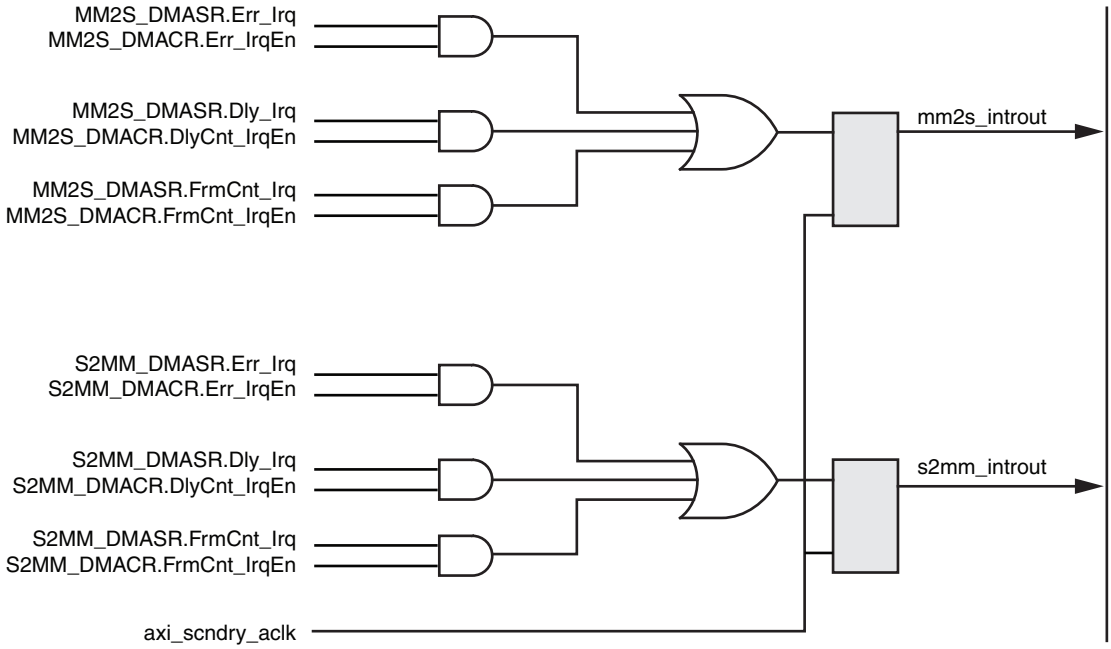


Figure 3-13: **Interrupt Out Concept**

Threshold Interrupt

Interrupt coalescing can be accomplished by setting the DMACR.IRQFrameCount field. With each frame completion event (that is, transmitted last line of frame or Received last line of frame when C_USE_FSYNC = 0 or on `mm2s_fsync` or `s2mm_fsync` for when C_USE_FSYNC = 1,2,3) the frame count is decremented. When the count reaches zero, FrmCnt_Irq asserts and if FrmCnt_IrqEn = 1, then an interrupt is generated on the associated channels introut signal (that is, `mm2s_introut` or `s2mm_introut`). When a frame count interrupt is generated, the frame count is reloaded in the counter in preparation for the next frame count event.

The internal frame count value is presented to software in DMASR.IRQFrameCntSts. A DMACR.IRQFrameCount value of 0x01 (default) cause a single frame count interrupt event to immediately generate an interrupt out.

If the delay interrupt is enabled (DMACR.IRQDelayCount not equal to 0 and DMACR.DlyIrq_En = 1), then a delay interrupt event also reloads the internal frame count counter. Finally with each software write of the frame count value (DMACR.IRQFrameCount), the internal counter is reloaded.

Figure 3-14 illustrates the functional composition of the interrupt threshold logic.

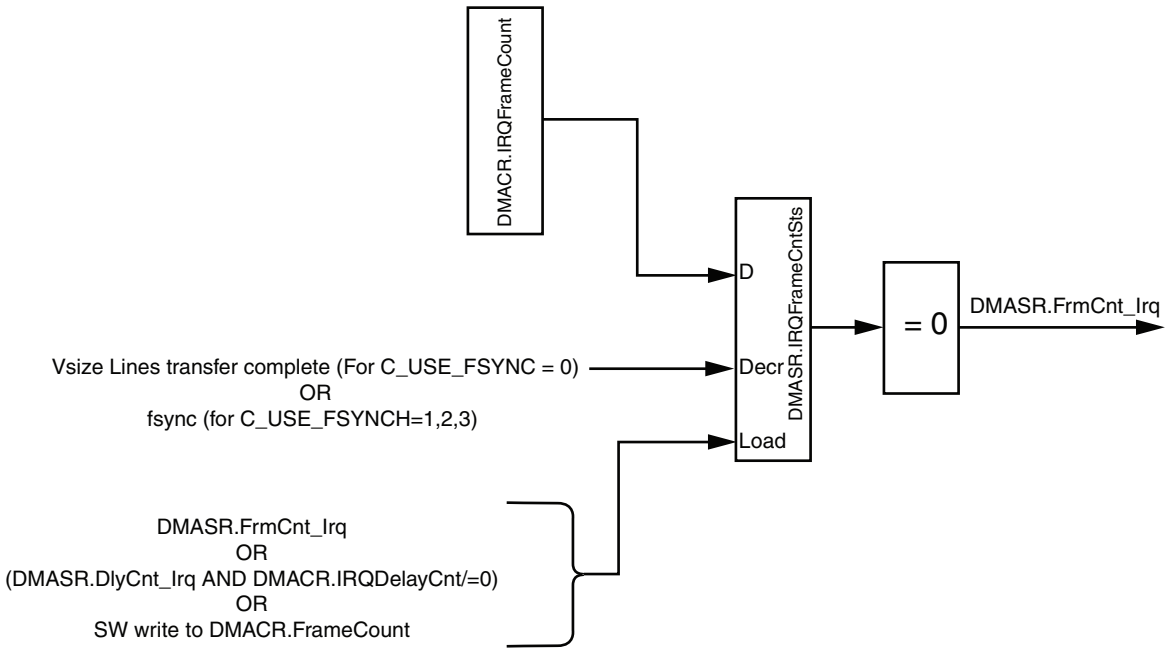


Figure 3-14: **Frame Count Interrupt Logic Concept**

Delay Interrupt

The delay interrupt is a mechanism by which software can receive an interrupt if there is a huge delay between `fsync` pulse assertion and start of packet (assertion of `tvalid`). Figure 3-15 shows a high-level block diagram of the delay interrupt architecture.

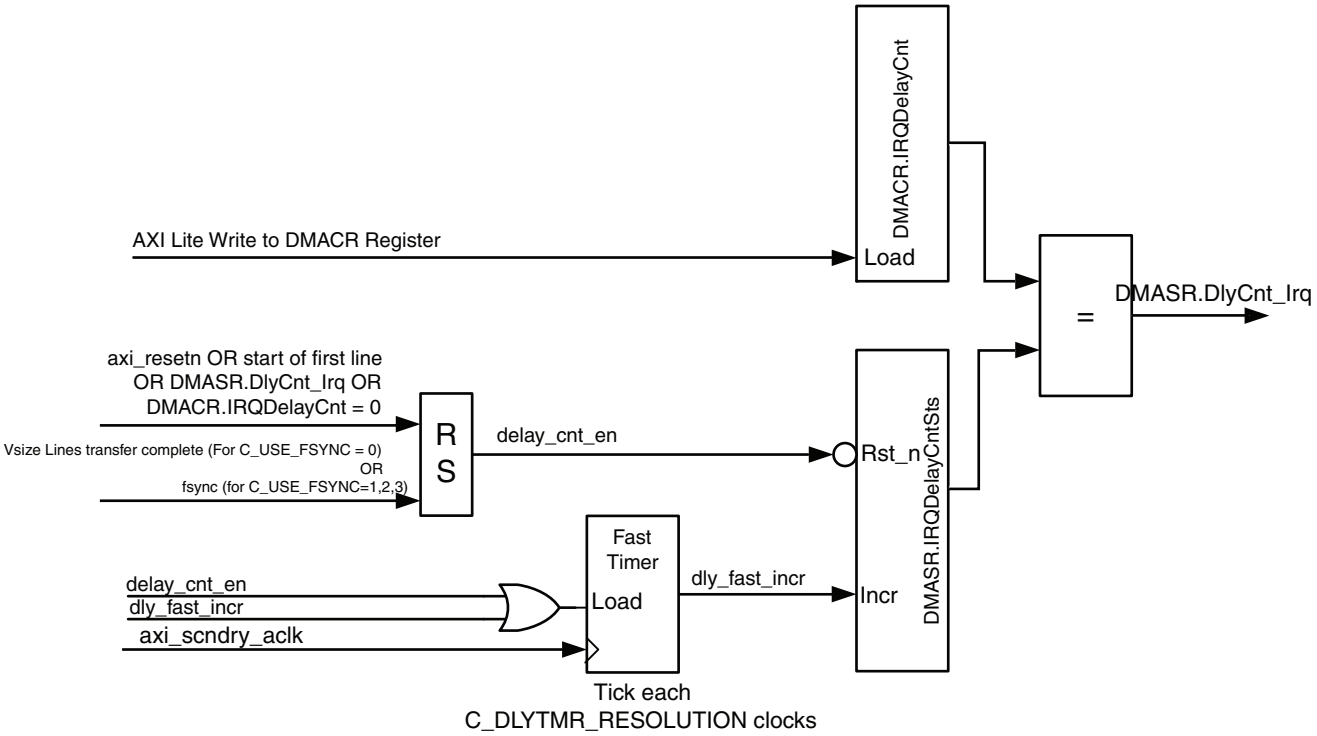


Figure 3-15: **Delay Interrupt Logic Concept**

The delay count interrupt is enabled by setting the DMACR.IRQDelay value to a non-zero value. The delay counter begins counting either upon receipt of frame sync (`mm2s_fsync` or `s2mm_fsync`) for C_USE_FSYNC = 1,2,3 or the completion of the transfer of vsize lines in free run mode for the respective channel. The delay counter resets with subsequent Start of Packet (`tvalid` assertion). When a delay interrupt event occurs, the delay timer is reset to zero, generating a IRQDelayCount event. If the DMACR.DlyCnt_IrqEn = 1 for the respective channel, then an interrupt out is generated from AXI VDMA. The delay timer does not count until the CPU services the interrupt by clearing the DMASR.DlyCnt_Irq bit to 0. Figure 3-16 shows example timing for this situation.

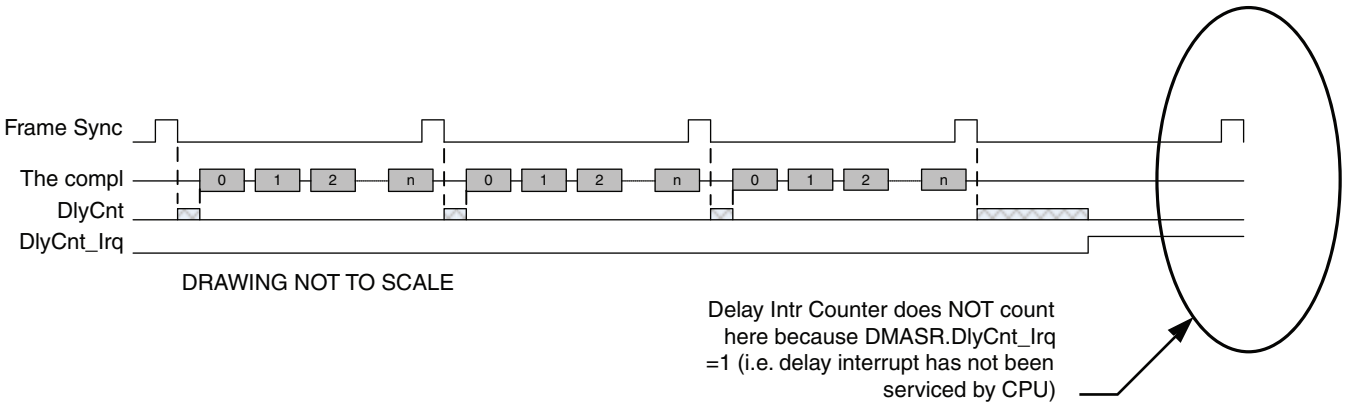


Figure 3-16: **Example Delay Timer Timing**

Errors

Any detected error on the primary datapath (that is, DMAIntErr, DMASlvErr, and DMADecErr) results in the associated channel (MM2S or S2MM) to halt gracefully when C_FLUSH_ON_FSYNC=0. Any Scatter Gather Engine detected error (that is, SGSlvErr or SGDecErr) causes the entire AXI VDMA engine to halt gracefully.

When an error is detected, the errored channel's DMACR.RS bit is set to 0. In accordance with AXI protocol all AXI transfers on memory map interfaces must complete; therefore, the AXI VDMA completes all pending transactions (transactions already posted and accepted on memory map interfaces) before setting the errored channel's DMASR.Halted bit. When the DMASR.Halted bit is set to 1, then the AXI VDMA channel is truly halted.

Furthermore for DMA detected errors the associated channel's FrmStore pointer (RdFrmStore or WrFrmStore) is updated with the frame reference of the errored frame. For SG detected errors the descriptor associated with the errored transfer is updated to the channel's CURDESC pointer register. If multiple, simultaneous errors are detected, then only one of the detected error's CURDESC is updated.

To resume operations, a reset must be issued, either soft or hard. If using soft reset, then for Scatter gather errors a reset must be issued to both channels. For primary DMA errors, the reset must be issued only to the channel that logged the error. Hard resets, reset the entire AXI VDMA engine.

Following is a list of possible errors:

DMAIntErr (C_INCLUDE_SG = 1/0)

DMA Internal Error flags an internal error in the AXI DataMover was detected. This can occur under two conditions.

This error can occur when a BTT = 0 is written to the primary AXI DataMover. For Scatter Gather Mode (C_INCLUDE_SG = 1) this happens if a descriptor is fetched with the VSIZE or HSIZE = 0. For Register Direct Mode (C_INCLUDE_SG = 0) this happens if the VSIZE and/or HSIZE register for the respective channel = 0 when the VSIZE register was written. This error also occurs when there is a frame size mismatch between programmed vsize and received(S2MM path)/transmitted(MM2S path) lines.

DMASlvErr (C_INCLUDE_SG = 1/0)

DMA Slave Error occurs when the slave to/from which data are transferred responds with a SLVERR on the memory map interface.

DMADecErr (C_INCLUDE_SG = 1/0)

DMA Decode Error occurs when the address request is targeted to an address that does not exist.

SGSlvErr (C_INCLUDE_SG = 1)

Scatter Gather Slave Error occurs when the slave from which descriptors are fetched responds with a SLVERR.

SGDecErr (C_INCLUDE_SG = 1)

Scatter Gather Decode Error occurs when the address request is targeted to an address that does not exist.

Table 3-11: **MM2S Errors**

Cause	MM2S Errors	Operating Mode	
	Register setting	**Free run(C_USE_FYNC = 0)**	**External fsync(C_USE_FSYNC=1) and Flush on fsync(C_FLUSH_ON_FSYNC=1)**
VDMA configured for too many lines w.r.t. Streaming Slave	SOFEarlyErr (bit7 of offset 0x04h). R/WC in flush mode. RO in non-flush mode.	This error cannot happen in free-run mode.	Channel does not HALT. Channel goes out-of-sync until next fsync, asserts TVALID low and does not provide any data until next fsync.
VDMA configured for too few lines w.r.t. Streaming Slave	Expect video IP to detect error	This error cannot happen in MM2S channel.	This error cannot happen in MM2S channel.
VDMA configured for too many bytes per line w.r.t. Streaming Slave	Expect video IP to detect error	This error cannot happen in MM2S channel.	This error cannot happen in MM2S channel.
VDMA configured for too few bytes per line w.r.t. Streaming Slave	Expect video IP to detect error	This error cannot happen in MM2S channel.	This error cannot happen in MM2S channel.

Table 3-12: **S2MM Errors**

Cause	S2MM Errors	Operating Mode	
	Register setting	**Free run(C_USE_FYNC = 0)**	**External fsync(C_USE_FSYNC=1) and Flush on fsync(C_FLUSH_ON_FSYNC=1)**
VDMA configured for too many lines w.r.t. Streaming Master	SOFEarlyErr (bit 7 of offset 0x34h). R/WC in flush mode. RO in non-flush mode.	This error cannot happen in free-run mode. VDMA waits infinitely for pending lines.	Channel does not HALT. Channel goes out-of-sync until next fsync, asserts TREADY high but does not transfer data till next fsync.
VDMA configured for too few lines w.r.t. Streaming Master	SOFLateErr (bit 11 of offset 0x34h). R/WC in flush mode. RO in non-flush mode.	This error cannot happen in free-run mode.	After receiving the VSIZE number of video lines, channel continues to assert TREADY high. If Streaming Master drives more data, it is dropped and this error bit is set.
VDMA configured for too many bytes per line w.r.t. Streaming Master	EOLEarlyErr (bit 8 of offset 0x34h). R/WC in all modes	Channel does not HALT.	Channel does not HALT.
VDMA configured for too few bytes per line w.r.t. Streaming Master	EOLLateErr (bit 15 of offset 0x34h). R/WC in all modes	Channel does not HALT. Extra bytes received in the line are dropped.	Channel does not HALT. Extra bytes received in the line are dropped.

Note: It is recommended to have C_MM2S_SOF_ENABLE = 1 C_S2MM_SOF_ENABLE = 1 and C_FLUSH_ON_FSYNC = 1 where all VDMA is continued to work irrespective of error conditions mentioned in the preceding tables. Setting C_S2MM_SOF_ENABLE = 1 also provides access to fsync cross-bar through which different fsync sources can be selected.

Triple Frame Buffer Example

Triple buffers or drop/add frame synchronizers are a common use for the AXI VDMA. Triple buffers can be used to pass image frames between two distinct clock domains without shear. Shear occurs when a frame is read from memory but that frame is made up of two different write frames. When the read and write clocks are asynchronous, the read and write memory pointers will cross each other at some point in time resulting in shear. To avoid shear, read and write pointers are not allowed to cross each other which results in frames being repeated or skipped. One or more AXI VDMAs can be used to implement this feature.

Generic triple buffers get the name by using three distinct frame stores to write and read data into and from memory. At no time is the read or write frame location allowed to overlap the other (for example, one cannot read and write from frame store #1 at the same time). This removes the possibility of shear in the read image that is possible in a double buffer scheme. Generic triple buffers use the following rules to ensure that reads and writes do not occur in the same frame store.

- Because there are three distinct frame stores and two possible operations (read and write), one frame store is always available and is denoted as the extra frame.
- When the write frame finishes, its next frame location is the extra frame location. The extra frame's location becomes the write frame's location. In essence, the locations swap between write and extra when the write completes. If the swap occurs twice prior to read frame completion, a skip (drop) frame occurs.
- When the read frame finishes, its next frame location is the newer (more recently written) of the extra frame or read frame. If the read frame is newer than the extra frame, a repeat (add) frame occurs.

The AXI VDMA mimics triple buffer capabilities by using Genlock synchronization. See Genlock Synchronization. Genlock mode is different than the generic triple buffer. In a generic triple buffer, the write frame must know the extra frame location(s) and read frame location. Genlock mode removes this restriction by implementing an N buffer with N distinct frame stores. N is an integer value greater than the maximum ratio of (readClock/writeClock) or (writeClock/readClock). For example, reading at a clock rate of 74.25 MHz and writing at a clock rate of 148.5 MHz gives a maximum ratio of 2 and requires N=3 distinct frame stores (C_NUM_FSTORES = 3). Reading at a clock rate of 148.5 MHz and writing at a clock rate of 74.25 MHz gives the same maximum ratio of 2 with the same requirement of N=3. Generic triple buffers work by reading the last full frame written. Genlock mode enhances this capability by allowing the user to read an arbitrary number of frames behind the write frame by only modifying the FRMDLY register. This can be extremely useful in applications such as deinterlacing or motion adaptive noise reduction.

The following sequence details the steps required to implement a triple buffer with the AXI VDMA.

1. Configure the core with the following parameters.
   a. Determine the value to use for Frame Stores. This integer value should be greater than the maximum ratio of read and write clocks.
   b. Enable
      i. Use Frame Sync
      ii. Asynchronous Clocks
      iii. Flush on Frame Sync
      iv. MM2S and S2MM Channel
      v. MM2S and S2MM Store and Forward
      vi. S2MM Frame Repeat on Error
   c. Disable
      i. Scatter Gather Engine.
   d. Set S2MM Genlock Mode to Master.
   e. Per user requirements
      i. Set stream data widths to user data sizes.
      ii. Set memory map data widths to match AXI Interconnect requirements. Suggest stream data width x 4.
      iii. Set maximum burst size. Suggest 256 for maximum throughput. (See the *AXI Reference Guide* (UG761) for more information on maximizing the AXI Interconnect performance.)
      iv. Line Buffer Depth. (Suggest next power of two greater than maximum burst size x 6.)
      v. Enable Start Of Frame on tuser(0). Select this option if the target peripherals implement the AXI4-Stream Video Protocol as described in the Video IP: AXI Feature Adoption section of the UG761 *AXI Reference Guide*.
2. Configure the write registers as Genlock master.
   a. Set S2MM_DMACR (30h) to 0x00000003. This enables run/stop and Circular_Park.
   b. Set S2MM_Start_Address 1 (ACh) through S2MM_Start_AddressN to their required locations. These locations can be static (based on maximum frame size) or dynamic (based on actual frame size).

c. Set S2MM_FRMDLY_STRIDE (A8h) to the appropriate value. FRMDLY is 0 for the Genlock master. STRIDE is the number of bytes per line.

d. Set S2MM_HSIZE (A4h) to the number of bytes per line.

e. Set S2MM_VSIZE (A0h) to the number of lines per frame. VSIZE must be set last and starts the S2MM VDMA transactions.

3. Configure the read registers as Genlock slave.

a. Set MM2S_DMACR (00h) to 0x0000000B. This enables run/stop, Circular_Park, and SyncEn (Genlock).

b. Set MM2S_Start_Address1 (5Ch) through MM2S_Start_AddressN to their required locations. These locations should match their S2MM_Start_Address counterparts.

c. Set MM2S_FRMDLY_STRIDE (58h) to the appropriate value. FRMDLY is 1 for the Genlock slave. STRIDE is the number of bytes per line.

d. Set MM2S_HSIZE (54h) to the number of bytes per line.

e. Set MM2S_VSIZE (50h) to the number of lines per frame. VSIZE must be set last and starts the MM2S VDMA transactions.

When changing frame sizes, the incoming frame size might not match the S2MM HSIZE and VSIZE registers and an error is noted on S2MM DMASR register DMAIntErr. The preceding example enables S2MM Frame Repeat on Error and the next frame is written to the same frame store location invalidating the bad frame. If the start addresses are based on maximum frame sizes, no changes are necessary to their registers. However, the S2MM HSIZE and VSIZE registers must be updated with the new frame size. VSIZE should be written last. After a full frame is written into memory, the MM2S HSIZE and VSIZE registers can be updated and are used with the new frame after Frame Sync occurs.

SECTION II: VIVADO DESIGN SUITE

Customizing and Generating the Core

Constraining the Core

Detailed Example Design

Customizing and Generating the Core

This chapter includes information on using Xilinx tools to customize and generate the core using the Vivado™ IP Catalog. For more information about the Vivado Design Suite, see the Vivado Design Suite - 2012.2 User Guides web page.

Vivado IP Catalog GUI Options

To access the AXI VDMA, do the following:

1. Open a project by selecting File > Open Project or create a new project by selecting **File > New Project**.
2. Open **IP Catalog** and choose **AXI Infrastructure/Video & Image Processing** in the **View by Function** pane.
3. 3. Double-click **AXI Video Direct Memory Access** to display the AXI VDMA GUI.

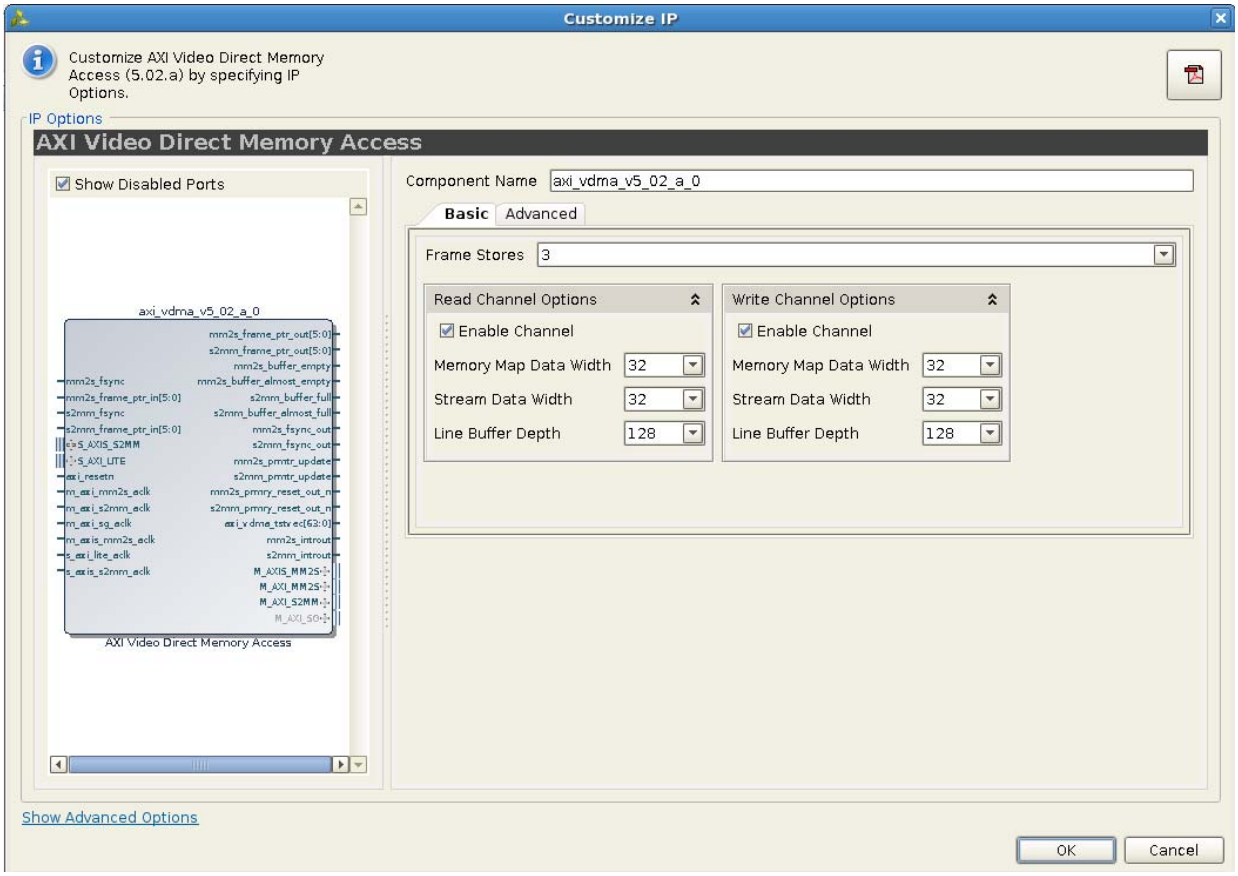


Figure 4-1: **Basic Options**

Basic Options

The following describes the fundamental options that affect both channels of the AXI VDMA core.

Frame Stores

This option enabled selection of the number of frame buffer storage locations to be processed by AXI VDMA.

Read Channel Options

It provides basic options for MM2S path.

Enable Channel

Checking this option enables MM2S Channel.

Memory Map Data Width

This option enables selection of desired AXI4 Memory Map data width for MM2S channel. Valid values are 32, 64, 128, 256, 512 and 1024.

Stream Data Width

This option enables selection of AXI4-Stream data width for MM2S channel. Valid values are multiples of 8 up to 1024 bits. This value must be less than or equal to Memory Map Data Width.

Line Buffer Depth

This option enables selection of line buffer depth for MM2S channel.

Write Channel Options

It provides basic options for S2MM path

Enable Channel

Checking this option enables S2MM Channel

Memory Map Data Width

This option enables selection desired AXI4 Memory Map data width for S2MM channel. Valid values are 32, 64, 128, 256, 512 and 1024.

Stream Data Width

This option enables selection of AXI4-Stream data width for S2MM channel. Valid values are multiples of 8 up to 1024 bits. This value must be less than or equal to Memory Map Data Width.

Line Buffer Depth

This option enables selection of line buffer depth for S2MM channel

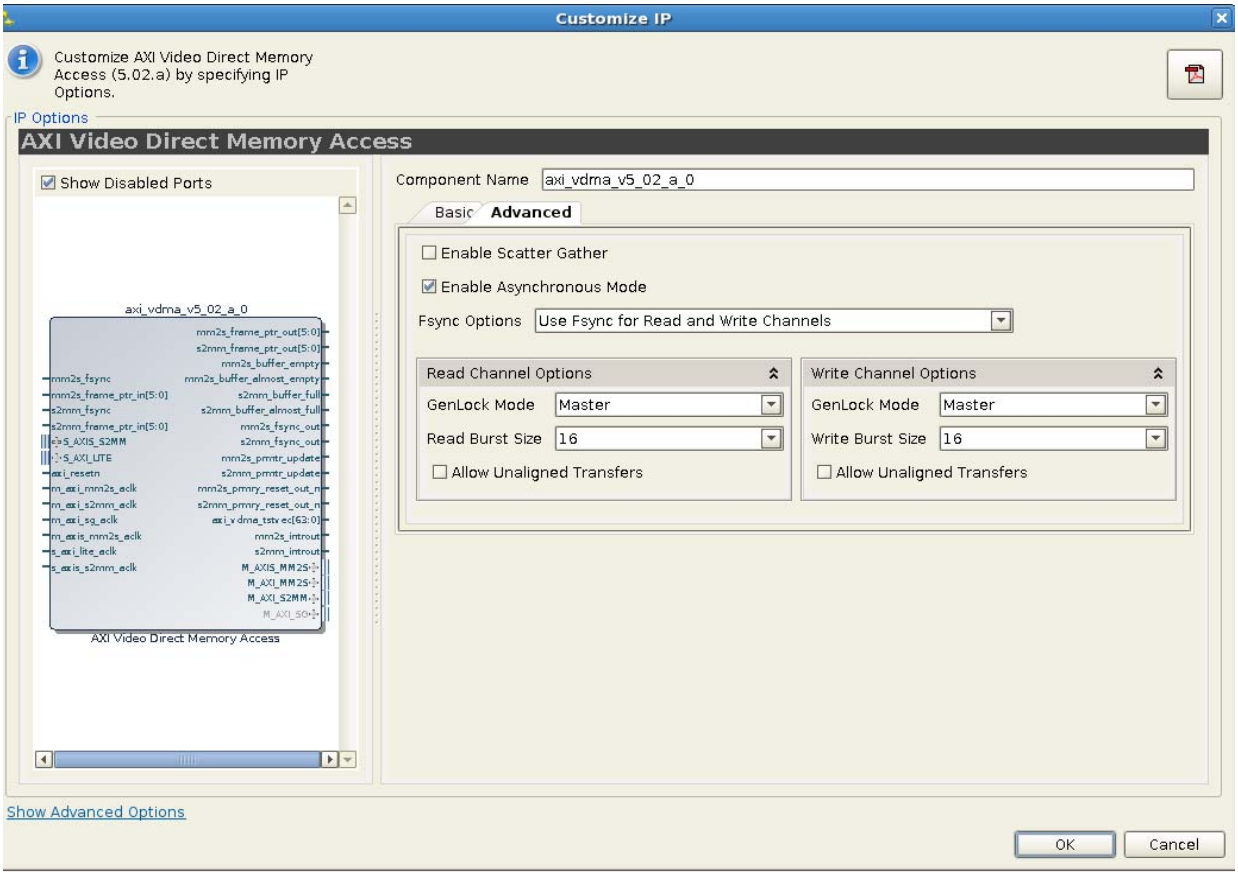


Figure 4-2: **Advanced Options**

Advanced Options

The following describes advanced options that affect both channels of the AXI VDMA core

Enable Scatter Gather

Checking this option enables Scatter Gather Mode operation and includes the Scatter Gather Engine in AXI VDMA. Unchecking this option enables Register Direct Mode operation, excluding the Scatter Gather Engine from AXI VDMA. Disabling the Scatter Gather Engine causes all output ports for the Scatter Gather engine to be driven zeros and input ports are ignored.

Enable Asynchronous Mode

This setting allows operation of the MM2S interface m_axi_mm2s_aclk, S2MM interface m_axi_s2mm_aclk, AXI4-Lite control interface s_axi_lite_aclk, and the Scatter Gather Interface m_axi_sg_aclk to be asynchronous 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.

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

Fsync Options

This option is used to set the synchronization mode of the AXI VDMA.

Read and Write Channels Free Running

Checking this option enables both MM2S (read) channel and S2MM(write) channel in free running mode.

Use Fsync for Read and Write Channels

Checking this option enables both MM2S(read) channel and S2MM(write) channel to synchronize with external fsync.

Use Fsync Only for Read Channel

Checking this option enables MM2S (read) channel to synchronize with external fsync and S2MM(write) channel is free running.

Use Fsync Only for Write Channel

Checking this option enables S2MM (read) channel to synchronize with external fsync and MM2S (write) channel is free running.

Read Channel Options

It provides basic options for MM2S path.

GenLock Mode

This option sets the Genlock Mode of the MM2S Channel. Selecting Master enables master mode and specifies that the MM2S channel operate as a Genlock Master. In Master mode, frames are not dropped or repeated. The current master frame being worked on by the MM2S channel is specified on the `mm2s_frm_ptr_out` port. Selecting Slave enables slave mode and specifies that the MM2S channel operate as a Genlock Slave. In Slave mode, frames are automatically dropped or repeated based on the master and slave frame rates.

The Genlock slave looks at the vector slice of `mm2s_frm_ptr_in` as specified in the MM2S DMACR Read Pointer Number field (DMACR.RdPntrNmbr bits 11 downto 8) to determine which frame the master is working on and operates a minimum Frame Delay behind the master.

Selecting Dynamic Master enables Genlock Master to dynamically skip the frame buffers that Slave is operating on. Dynamic Master outputs previously written frame pointer on `mm2s_frm_ptr_out`. It also samples the value on `mm2s_frm_ptr_in` to switch to the appropriate frame buffer. Selecting Dynamic Slave enables Genlock Slave to work on the latest frame that the Master has operated on. Frame Delay is not valid in Dynamic Genlock modes. See C_MM2S_GENLOCK_MODE in Parameter Descriptions for more details.

Read Burst Size

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

Allow Unaligned Transfers

Enables or disables the MM2S Data Realignment Engine. When checked, the data realignment engine is enabled and allows data realignment to the byte (8 bits) level on the MM2S Memory Map datapath. The MM2S channel reads the vertical size (vsize) number of video lines each horizontal size (hsize) bytes long and spaced stride bytes apart (stride is number of bytes between first pixel of each line) from memory. For the case where unaligned transfers are allowed, data reads can start from any Start Address byte offset and be of any horizontal size and stride value. The read data are aligned such that the first byte read is the first valid byte out on the AXI4-Stream.

When unchecked, that is, for the case where unaligned transfers are not allowed, the Start Address must be aligned to multiples of C_M_AXI_MM2S_DATA_WIDTH bytes. Also Horizontal Size and Stride must be specified in even multiples of C_M_AXI_MM2S_DATA_WIDTH bytes. For example, if C_M_AXI_MM2S_DATA_WIDTH = 32, data is aligned if the Start Address at word offsets (32-bit offset), that is, 0x0, 0x4, 0x8, 0xC, and so on., Horizontal Size is 0x4, 0x8, 0xC and so on. Stride is 0x4, 0x8, 0xC, and so on. If C_M_AXI_MM2S_DATA_WIDTH = 64, data is aligned if the Start Address is at double-word offsets (64-bit offsets), that is, 0x0, 0x8, 0x10, 0x18, and so on, and Horizontal Size, and Stride are at 0x4, 0x8, 0xC, and so on.

Note: If Allow Unaligned Transfers is unchecked, unaligned start addresses, hsizes, or strides, are not supported. Having an unaligned Start Address, HSize, and/or Stride results in undefined behavior.

Note: Further the Data Realignment Engine only supports AXI4-Stream data width setting of 64-bits and less.

Write Channel Options

It provides basic options for S2MM path.

Genlock Mode

This option sets the Genlock Mode of the S2MM Channel. Selecting Master enables master mode and specifies that the S2MM channel operate as a Genlock Master. In Master mode, frames are not dropped or repeated. The current master frame being worked on by the S2MM channel is specified on the `s2mm_frm_ptr_out` port. Selecting Slave enables slave mode and specifies that the S2MM channel operate as a Genlock Slave. In Slave mode, frames are automatically dropped or repeated based on the master and slave frame rates. The Genlock slave looks at the vector slice of `s2mm_frm_ptr_in` as specified in the S2MM DMACR Write Pointer Number field (DMACR.WrPntrNmbr bits 11 downto 8) to determine which frame the master is working on and operates a minimum Frame Delay behind the master.

Selecting Dynamic Master enables Genlock Master to dynamically skip the frame buffers that Slave is operating on. Dynamic Master outputs previously written frame pointer on `s2mm_frm_ptr_out`. It also samples the value on `s2mm_frm_ptr_in` to switch to the appropriate frame buffer. Selecting Dynamic Slave enables Genlock Slave to work on the latest frame that the Master has operated on. Frame Delay is not valid in Dynamic Genlock modes. See C_S2MM_GENLOCK_MODE in Parameter Descriptions for more details.

Write Burst Size

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

Allow Unaligned Transfers

Enables or disables the S2MM Data Realignment Engine. When checked, the data realignment engine is enabled and allows data realignment to the byte (8 bits) level on the S2MM Memory Map datapath. For the case where Unaligned transfers are allowed, data writes can target any Start Address byte offset, be of any horizontal size and stride value; the write data is aligned such that the first byte received on AXI4-Stream is the first valid byte written to the specified memory offset. When unchecked, that is, for the case where unaligned transfers are not allowed, the Start Address must be aligned to multiples of C_M_AXI_S2MM_DATA_WIDTH bytes. Also Horizontal Size and Stride must be specified in even multiples ofC_M_AXI_S2MM_DATA_WIDTH bytes.

For example, if C_M_AXI_S2MM_DATA_WIDTH = 32, data are aligned if the Start Address at word offsets (32-bit offset), that is, 0x0, 0x4, 0x8, 0xC, and so on, Horizontal Size is 0x4, 0x8, 0xC and so on, Stride is 0x4, 0x8, 0xC, and so on. If C_M_AXI_S2MM_DATA_WIDTH = 64, data are aligned if the Start Address is at double-word offsets (64-bit offsets), that is, 0x0, 0x8, 0x10, 0x18, and so on, and Horizontal Size, and Stride are at 0x4, 0x8, 0xC, and so on.

Note: If Allow Unaligned Transfers is unchecked, unaligned start addresses, hsizes, or strides, are not supported. Having an unaligned Start Address, HSize, and/or Stride results in undefined behavior.

Note: Further, the Data Realignment Engine only supports AXI4-Stream data width setting of 64-bits and less.

For the RTL parameters which are hidden in the Vivado IP Catalog GUI, following are the example commands to get or set the value of a hidden RTL parameter.

```
get_property -name CONFIG.c_include_mm2s_sf -object [get_ips axi_vdma_v5_02_a_0] 0

set_property -name CONFIG.c_include_mm2s_sf -value {1} -objects [get_ips
axi_vdma_v5_02_a_0] 1

get_property -name CONFIG.c_include_mm2s_sf -object [get_ips axi_vdma_v5_02_a_0] 1
```

Output Generation

The output hierarchy when the core is generated from Vivado IP Catalog is shown in Figure 4-3:

The component name of the IP generated is axi_vdma_v5_02_a_0.

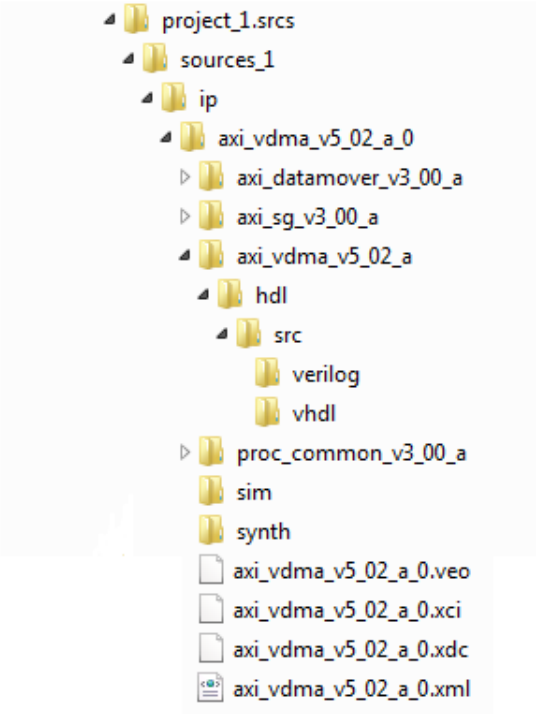


Figure 4-3: **Output Hierarchy**

Vivado IP Catalog delivers the RTL files of the axi_vdma core as well as all its helper cores. The axi_datamover_v3_00_a, axi_sg_v3_00_a, and proc_common_v3_00_a are the helper cores used by the axi_vdma. The RTL files of axi_vdma are delivered under /ip/axi_vdma_v5_02_a/hdl/src/vhdl. The sim and synth folders contain the wrappers for simulation and synthesis respectively. The tool also delivers the instantiation template file .veo/.vho.

Constraining the Core

In synchronous mode, C_PRMRY_IS_ACLK_ASYNC = 0, all clocks run at the same frequency and are derived from the same source. There are no multicycle or false paths in this design. All logic between flop-to-flop should meet timing within one clock period.

In asynchronous mode, C_PRMRY_IS_ACLK_ASYNC = 1, all clocks are treated asynchronously to each other and the core will write out appropriate clock domain crossing constraints.

Detailed Example Design

There an no applicable example designs for the Vivado™ Design Suite.

SECTION III: ISE DESIGN SUITE

Customizing and Generating the Core

Constraining the Core

Detailed Example Design

Chapter 7

Customizing and Generating the Core

Generating the Core Using CORE Generator Tool

The AXI VDMA can be found in **AXI Infrastructure/Video & Image Processing** in the CORE Generator™ tool graphical user interface (GUI) **View by Function** pane.

To access the AXI VDMA, do the following:

1. Open a project by selecting **File** > **Open Project** or create a new project by selecting **File** > **New Project**.
2. With an open project, choose **AXI Infrastructure/Video & Image Processing** in the **View by Function** pane.
3. Double-click **AXI Video Direct Memory Access** to display the AXI VDMA GUI.

CORE Generator Tool Parameter Screen

The AXI VDMA GUI contains one screen (Figure 7-1) that provides information about the core, allows for configuration of the core, and provides the ability to generate the core.

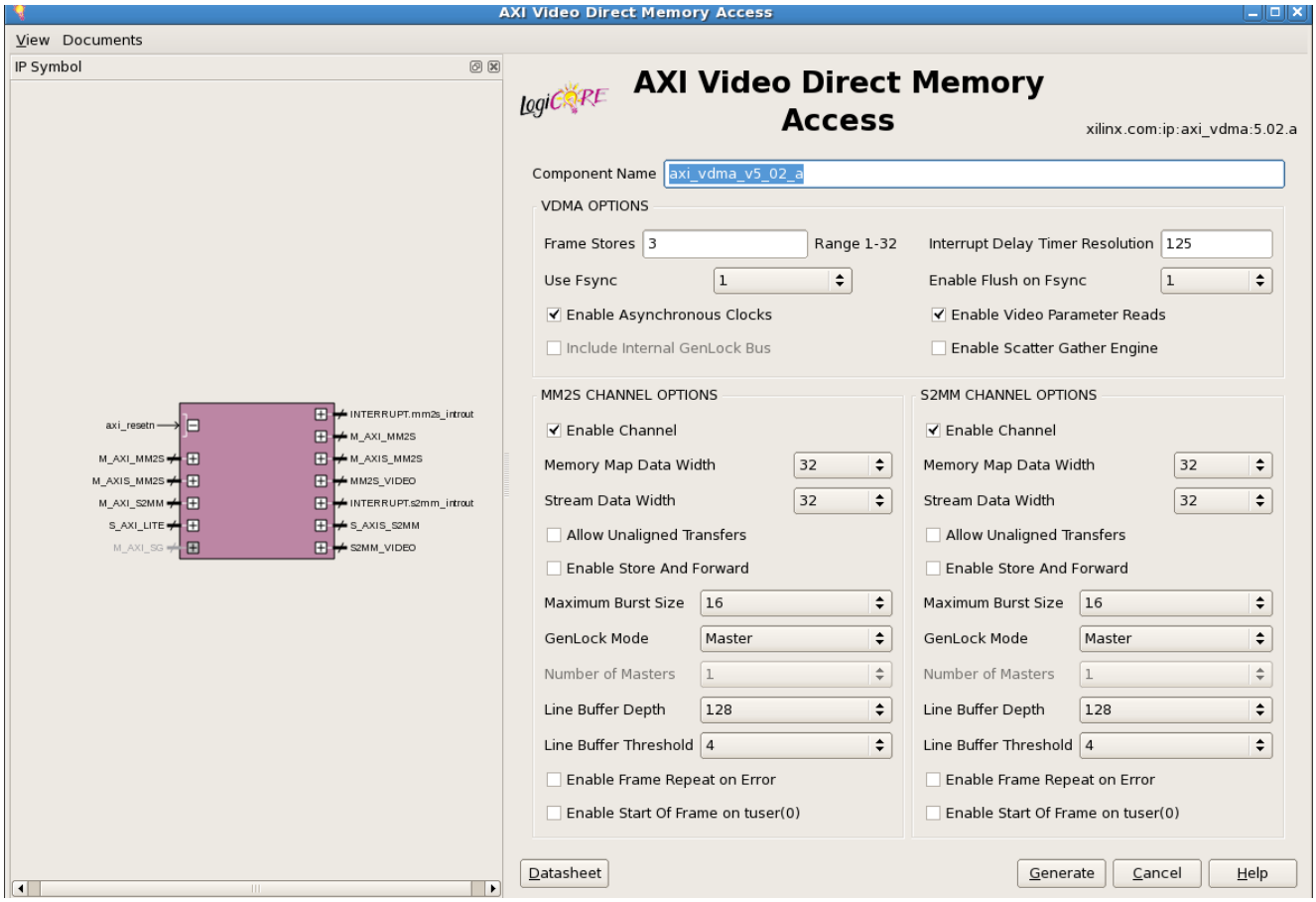


Figure 7-1: **AXI VDMA GUI**

Component Name

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

VDMA Options

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

Frame Stores

Frame Stores indicates the number of frame buffer storage locations to be processed by the AXI VDMA. In Register Direct Mode (C_INCLUDE_SG = 0) this value determines the number of valid Start Addresses per channel that need to be initialized. For Scatter Gather mode (C_INCLUDE_SG = 1) this parameter defines the number of Scatter Gather descriptors per channel in the descriptor chain required to initialize the AXI VDMA. Valid values are 1 to 32.

Use Frame Sync

This option is used to set the synchronization mode of the AXI VDMA. Uncheck to select Free Run Mode and check to select Frame Sync Mode. In Free Run Mode the AXI VDMA transfers data as quickly as it is able to. When in Frame Sync Mode, the AXI VDMA transfers data starting with the falling edge of each `mm2s_fsync` or `s2mm_fsync` for the associated channel. There are options to select Frame Sync mode for MM2S and S2MM channels independently. See C_USE_FSYNC in Parameter Descriptions for more details.

Enable Scatter Gather Engine

Checking this option enables Scatter Gather Mode operation and includes the Scatter Gather Engine in AXI VDMA. Unchecking this option enables Register Direct Mode operation, excluding the Scatter Gather Engine from AXI VDMA. Disabling the Scatter Gather Engine causes all output ports for the Scatter Gather engine to be driven zeros and input ports are ignored.

Enable Video Parameter Reads

Checking this option enables the reading of the video transfer parameters (vsize, hsize, stride, and frame delay) and start addresses using the `s_axi_lite` control interface. For applications where reading of the video transfer parameters is not needed then unchecking this option disables reading of video transfer parameters which saves FPGA resources.

Interrupt Delay Timer Resolution

This integer value sets the resolution of the Interrupt Delay Counter. Values specify the number of clock cycles between each tick of the delay counter. If Scatter Gather Engine is enabled, clock cycles are based on the `m_axi_sg_aclk` clock input. If Scatter Gather Engine is disabled, clock cycles are based on `s_axi_lite_aclk` clock cycles.

Enable Asynchronous Clocks

This setting allows operation of the MM2S interface `m_axi_mm2s_aclk`, S2MM interface `m_axi_s2mm_aclk`, AXI4-Lite control interface `s_axi_lite_aclk`, and the Scatter Gather Interface `m_axi_sg_aclk` to be asynchronous 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`. When Asynchronous Clocks are disabled, all clocks must be at the same frequency and from the same source.

Enable Flush on Frame Sync

This setting enables the AXI VDMA to reset internal states and flush transfer data on frame sync. This allows VDMA to restart transfers at the beginning of a new frame after DMA Internal error detection as opposed to halting. This feature is only enabled when the channel uses external frame sync. There are options to enable flush on frame sync mode for MM2S and S2MM channels independently.

Include Internal Genlock Bus

This setting allows internal routing of MM2S and S2MM Genlock buses without having connecting them outside the core.

MM2S Channel Options

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

Enable Channel

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

Memory Map Data Width

Data width in bits of the AXI MM2S Memory Map Read data bus. Valid values are 32, 64, 128, 512 and 1024.

Stream Data Width

Data width in bits of the AXI MM2S AXI4-Stream data bus. Valid values are multiples of 8 up to 1024 bits. This value must be less than or equal to Memory Map Data Width.

Allow Unaligned Transfers

Enables or disables the MM2S Data Realignment Engine. When checked, the data realignment engine is enabled and allows data realignment to the byte (8 bits) level on the MM2S Memory Map datapath. The MM2S channel reads the vertical size (vsize) number of video lines each horizontal size (hsize) bytes long and spaced stride bytes apart (stride is number of bytes between first pixel of each line) from memory. For the case where unaligned transfers are allowed, data reads can start from any Start Address byte offset, be of any horizontal size and stride value. The read data are aligned such that the first byte read is the first valid byte out on the AXI4-Stream.

When unchecked, that is, for the case where unaligned transfers are *not* allowed, the Start Address must be aligned to multiples of C_M_AXI_MM2S_DATA_WIDTH bytes. Also Horizontal Size and Stride must be specified in even multiples of C_M_AXI_MM2S_DATA_WIDTH bytes.

For example, if C_M_AXI_MM2S_DATA_WIDTH = 32, data is aligned if the Start Address at word offsets (32-bit offset), that is, 0x0, 0x4, 0x8, 0xC, and so on., Horizontal Size is 0x4, 0x8, 0xC and so on. Stride is 0x4, 0x8, 0xC, and so on.

If C_M_AXI_MM2S_DATA_WIDTH = 64, data is aligned if the Start Address is at double-word offsets (64-bit offsets), that is, 0x0, 0x8, 0x10, 0x18, and so on, and Horizontal Size, and Stride are at 0x4, 0x8, 0xC, and so on.

Note: If Allow Unaligned Transfers is unchecked then unaligned start addresses, hsizes, or strides, are not supported. Having an unaligned Start Address, HSize, and/or Stride results in undefined behavior.

Note: Further the Data Realignment Engine only supports AXI4-Stream data width setting of 64-bits and less.

Enable Store and Forward

This option enables or disables the Store and Forward buffer for the MM2S channel. When enabled, read requests on MM2S are only made if there is enough buffer space in the Store-and-Forward buffer to complete the burst. When MM2S Line Buffer Depth is not zero and Store and Forward is enabled, the stream valid signal, `m_axis_mm2s_tvalid`, does not assert until a minimum Line Buffer Threshold bytes have been read and stored in the Store-And-Forward buffer.

Maximum Burst Size

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

Genlock Mode

This option sets the Genlock Mode of the MM2S Channel. Selecting **Master** enables master mode and specifies that the MM2S channel operate as a Genlock Master. In Master mode, frames are not dropped or repeated. The current master frame being worked on by the MM2S channel is specified on the `mm2s_frm_ptr_out` port. Selecting **Slave** enables slave mode and specifies that the MM2S channel operate as a Genlock Slave. In Slave mode, frames are automatically dropped or repeated based on the master and slave frame rates.

The Genlock slave looks at the vector slice of mm2s_frm_ptr_in as specified in the MM2S DMACR Read Pointer Number field (DMACR.RdPntrNmbr bits 11 downto 8) to determine which frame the master is working on and operates a minimum Frame Delay behind the master.

Selecting **Dynamic Master** enables Genlock Master to dynamically skip the frame buffers that Slave is operating on. Dynamic Master outputs previously written frame pointer on mm2s_frm_ptr_out. It also samples the value on mm2s_frm_ptr_in to switch to appropriate frame buffer. Selecting **Dynamic Slave** enables Genlock Slave to work on the latest frame that the Master has operated on. Frame Delay is not valid in Dynamic Genlock modes. See C_MM2S_GENLOCK_MODE in Parameter Descriptions for more details.

Number of Masters

This setting specifies to the Genlock slave the total number of masters to synchronize operations to. This setting also specifies the vector width of the mm2s_frm_ptr_in port, where each master requires 6 bits on the mm2s_frm_ptr_in vector. Therefore, the width of the mm2s_frm_ptr_in port is 6*Number of Masters. Valid values are 1 to 16.

Line Buffer Depth

This setting specifies the inclusion of an MM2S Line Buffer and also specifies the depth. A setting of zero excludes the line buffer from the MM2S Channel. A non-zero value includes the Line Buffer and sets the depth in bytes of the line buffer. The line buffer resides on the MM2S AXI4-Stream Interface. Valid minimum depth, excluding 0, equals C_M_AXIS_MM2S_TDATA_WIDTH/8, must always be a power of 2 value. In case this division produces a non-power of 2 value, the allowed minimum depth is nearest to the upper power of 2 value. See C_MM2S_LINEBUFFER_DEPTH in Parameter Descriptions for more details.

Line Buffer Threshold

This specifies the almost full threshold value of the MM2S_THRESHOLD register at which the almost full flag asserts/deasserts. This value is ignored by AXI VDMA if the Line Buffer Depth is set to 0. This value must be a resolution of AXI4-Stream data width in bytes (C_M_AXIS_MM2S_TDATA_WIDTH/8), with a minimum setting of C_M_AXIS_MM2S_TDATA_WIDTH/8 and a maximum setting of Line Buffer Depth (C_MM2S_LINEBUFFER_DEPTH).

Note: If C_M_AXIS_MM2S_TDATA_WIDTH is a non-power of 2, then the line buffer threshold value should be calculated based on the nearest upper power of 2 value. For example, if C_M_AXIS_MM2S_TDATA_WIDTH = 24, then the threshold values should be calculated based on the nearest upper power of 2. That is, C_M_AXIS_MM2S_TDATA_WIDTH = 32. See C_MM2S_LINEBUFFER_THRESH in Parameter Descriptions for more details.

Enable Frame Advancement on Error

This setting enables or disables the MM2S Channel frame advancement on error when the channel is selected and operating as a master and Flush on Frame Sync setting is enabled. When an error is detected in a particular frame, this setting allows the user to let the frame number advance on the next frame sync or not advance and re-use the errored frame's frame number. This is used in applications where it is desired to hide the errored frame.

Enable Start Of Frame on tuser(0)

This setting enables SOF generation for MM2S channel. SOF pulse is driven on m_axis_mm2s_tuser(0) coincident with first pixel of first line for each frame. For additional information, see the Video IP: AXI Feature Adoption section of the UG761 *AXI Reference Guide*.

S2MM Channel Options

The following subsections describe options that affect only the S2MM Channel of the AXI VDMA core.

Enable Channel

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

Clock Frequency

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

Memory Map Data Width

Data width in bits of the AXI S2MM Memory Map Write data bus. Valid values are 32, 64, 128, 512 and 1024.

Stream Data Width

Data width in bits of the AXI S2MM AXI4-Stream Data bus. Valid values are multiples of 8 up to 1024 bits. This value must be less than or equal to Memory Map Data Width.

Allow Unaligned Transfers

Enables or disables the S2MM Data Realignment Engine. When checked, the data realignment engine is enabled and allows data realignment to the byte (8 bits) level on the S2MM Memory Map datapath. For the case where Unaligned transfers are allowed, data writes can target any Start Address byte offset, be of any horizontal size and stride value; the write data is aligned such that the first byte received on AXI4-Stream is the first valid byte written to the specified memory offset.

When unchecked, that is, for the case where unaligned transfers are *not* allowed, the Start Address must be aligned to multiples of C_M_AXI_S2MM_DATA_WIDTH bytes. Also Horizontal Size and Stride must be specified in even multiples of C_M_AXI_S2MM_DATA_WIDTH bytes.

For example, if C_M_AXI_S2MM_DATA_WIDTH = 32, data are aligned if the Start Address at word offsets (32-bit offset), that is, 0x0, 0x4, 0x8, 0xC, and so on, Horizontal Size is 0x4, 0x8, 0xC and so on, Stride is 0x4, 0x8, 0xC, and so on.

If C_M_AXI_S2MM_DATA_WIDTH = 64, data are aligned if the Start Address is at double-word offsets (64-bit offsets), that is, 0x0, 0x8, 0x10, 0x18, and so on, and Horizontal Size, and Stride are at 0x4, 0x8, 0xC, and so on.

Note: If Allow Unaligned Transfers is unchecked, then unaligned start addresses, hsizes, or strides, are not supported. Having an unaligned Start Address, HSize, and/or Stride results in undefined behavior.

Note: Further, the Data Realignment Engine only supports AXI4-Stream data width setting of 64-bits and less.

Enable Store and Forward

This option enables or disables the Store and Forward buffer for the S2MM channel. When enabled, writes are only requested if all of the write data to complete the burst is stored in the Store-and-Forward buffer.

Note: On S2MM if data bus upsizing is required, that is, Stream Data Width does not equal Memory Map Data Width, then throttles (`m_axi_s2mm_wvalid` = 0) between data beat writes are observed during the packing processes. For example, if Stream Data Width = 16 and Memory Map Data Width = 32 then throttles occur every 2 clocks. The maximum throttle case would be when the Stream Data Width = 8 and Memory Map Data Width = 256 giving a 32 clock throttle between data beats.

Maximum Burst Size

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

Genlock Mode

This option sets the Genlock Mode of the S2MM Channel. Selecting **Master** enables master mode and specifies that the S2MM channel operate as a Genlock Master. In Master mode, frames are not dropped or repeated. The current master frame being worked on by the S2MM channel is specified on the `s2mm_frm_ptr_out` port. Selecting **Slave** enables slave mode and specifies that the S2MM channel operate as a Genlock Slave. In Slave mode, frames are automatically dropped or repeated based on the master and slave frame rates. The Genlock slave looks at the vector slice of `s2mm_frm_ptr_in` as specified in the S2MM DMACR Write Pointer Number field (DMACR.WrPntrNmbr bits 11 downto 8) to determine which frame the master is working on and operates a minimum Frame Delay behind the master.

Selecting **Dynamic Master** enables Genlock Master to dynamically skip the frame buffers that Slave is operating on. Dynamic Master outputs previously written frame pointer on `s2mm_frm_ptr_out`. It also samples the value on `s2mm_frm_ptr_in` to switch to the appropriate frame buffer. Selecting **Dynamic Slave** enables Genlock Slave to work on the latest frame that the Master has operated on. Frame Delay is not valid in Dynamic Genlock modes. See C_S2MM_GENLOCK_MODE in Parameter Descriptions for more details.

Number of Masters

This setting specifies to the Genlock slave the total number of masters to synchronize operations to. This setting also specifies the vector width of the `s2mm_frm_ptr_in` port, where each master requires 5 bits on the `s2mm_frm_ptr_in` vector. Therefore the width of the `s2mm_frm_ptr_in` port is 5*Number of Masters. Valid values are 1 to 16.

Line Buffer Depth

This specifies the inclusion of an S2MM Line Buffer and also specifies the depth. A setting of zero excludes the line buffer from the S2MM Channel. A non-zero value includes the Line Buffer and sets the depth in bytes of the line buffer. The line buffer resides on the S2MM AXI4-Stream Interface. Valid minimum depth, excluding 0, equals C_S_AXIS_S2MM_TDATA_WIDTH/8, must always be a power of 2 value. In case this division produces a non-power of 2 value, the allowed minimum depth is the nearest upper power of 2 value. See C_S2MM_LINEBUFFER_DEPTH in Parameter Descriptions for more details.

Line Buffer Threshold

This specifies the almost full threshold value of the S2MM_THRESHOLD register at which the almost full flag asserts/deasserts. This value is ignored by AXI VDMA if the Line Buffer Depth is set to 0. This value must be a resolution of the AXI4-Stream data width in bytes (C_S_AXIS_S2MM_TDATA_WIDTH/8), with a minimum setting of C_S_AXIS_S2MM_TDATA_WIDTH/8 and a maximum setting of Line Buffer Depth (C_S2MM_LINEBUFFER_DEPTH).

Note: If C_S_AXIS_S2MM_TDATA_WIDTH is a non-power of 2, the line buffer threshold value should be calculated based on the nearest upper power of 2 value. For example if C_S_AXIS_S2MM_TDATA_WIDTH = 24, the threshold values should be calculated based on the nearest upper power of 2. That is, C_S_AXIS_S2MM_TDATA_WIDTH = 32. See C_S2MM_LINEBUFFER_THRESH in Parameter Descriptions for more details.

Enable Start Of Frame on tuser(0)

This setting along with FsyncSrcSelect = 10 enables SOF detection for S2MM channel on `s_axis_s2mm_tuser(0)`. SOF pulse received on `s_axis_s2mm_tuser(0)` coincident with the first pixel of the first line for each frame. For additional information, see the Video IP: AXI Feature Adoption section of the UG761 *AXI Reference Guide*.

Enable Frame Advancement on Error

This setting enables or disables the S2MM Channel frame advancement on error when the channel is selected and operating as a master and Flush on Frame Sync setting is enabled. When an error is detected in a particular frame, this setting allows the user to let the frame number advance on the next frame sync or not advance and re-use the errored frame's frame number. This setting is used in applications where it is desired to hide the errored frame.

Generating the Core Using EDK

The AXI VDMA can be found in **IP Catalog - EDK_Install/DMA and Timer** in the Xilinx Platform Studio tool graphical user interface (GUI).

To access the AXI VDMA, do the following:

1. Invoke Xilinx Platform Studio and open a project by selecting **File > Open Project** or create a new project by selecting **File > New Project**.
2. With an open project, choose **EDK_Install/DMA and Timer**.
3. Double-click **AXI Video DMA** to display the AXI VDMA GUI.

For a new project:

1. Invoke Xilinx Platform Studio and create New Project using Base System Builder.
2. Select Interconnect Type (AXI System) and then select Board Name (based on 6/7 series FPGA)
3. After BSB is created, add AXI VDMA from IP catalog (**EDK_Install/DMA and Timer**) by double clicking **AXI Video DMA 5.01.a**. This opens up an EDK GUI which is described in the next section.

EDK pCore GUI

The AXI VDMA EDK GUI provides information about the core, allows for configuration of the core, and provides the ability to generate the core. The pCore is generated with each option set to the default value. Figure 7-2 illustrates the EDK pCore GUI for the AXI VDMA. All of the options in the EDK pCore GUI correspond to the same options in the CORE Generator tool GUI.

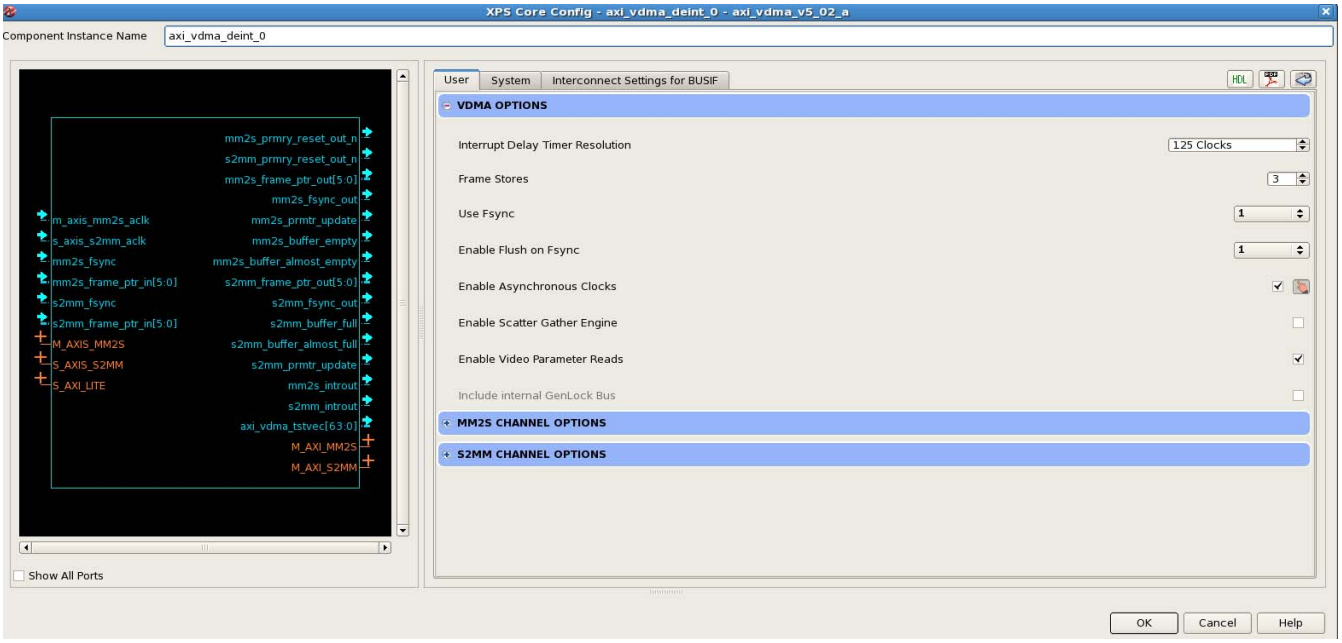


Figure 7-2: **EDK pCore GUI**

Output Generation

The output files generated from the CORE Generator tool are placed in the project directory. The file output list can include some or all of the following files.

Directory Hierarchy

<project directory>

Top-level project directory for the CORE Generator tool

 <project directory>/<axi_vdma_component name>

Contains the AXI VDMA doc and source files.

 <axi_vdma_component name>/doc

Contains the AXI VDMA solution PDF documentation.

 <axi_vdma_component name>/hdl/src/vhdl

Contains the source files for AXI VDMA core.

File Details

<project directory>

This is the top-level file. It contains templates for instantiation the core and the xco file.

Table 7-1: **Project Directory**

Name	Description
<project_dir>	
<axi_vdma_component_name>.xco	Log file from CORE Generator tool describing which options were used to generate the AXI VDMA core. An XCO file is generated by the CORE Generator tool for each core that it creates in the current project directory. An XCO file can also be used as an input to the CORE Generator tool.
<axi_vdma_component_name>_flist.txt	A text file listing all of the output files produced when the customized AXI VDMA core was generated in the CORE Generator tool.
<axi_vdma_component_name>.vho	The HDL template for instantiating the AXI VDMA core.
<axi_vdma_component_name_synth>.vhd	The HDL synthesis wrapper file with the modified parameter configuration of AXI VDMA core.
<axi_vdma_component_name_sim>.vhd	The structural simulation model for the AXI VDMA core. It is used for functionally simulating the core.

Back to Top

<project directory>/<axi_vdma_component name>

This directory contains the doc and hdl folder.

<axi_vdma_component name>/doc

This directory contains the appropriate product guide.

Table 7-2: **Doc Directory**

Name	Description
<project_dir>/<axi_vdma_component_name>/doc	
axi_vdma_v5_02_a_readme.txt axi_vdma_v5_02_a_readme_vinfo.html	The AXI VDMA release notes and core information file in text and html format.
pg020_axi_vdma.pdf	The AXI VDMA Product Guide.

Back to Top

<axi_vdma_component name>/hdl/src/vhdl

This directory contains AXI VDMA source files including the proc_common, AXI SG, AXI Data Mover helper library files.

Constraining the Core

In synchronous mode, C_PRMRY_IS_ACLK_ASYNC = 0, all clocks run at the same frequency and are derived from the same source. There are no multicycle or false paths in this design. All logic between flop-to-flop should meet timing within one clock period.

In asynchronous mode, C_PRMRY_IS_ACLK_ASYNC = 1, all clocks are treated asynchronously to each other and the core will write out appropriate clock domain crossing constraints.

Detailed Example Design

For detailed information about available example designs for the VDMA core, see the Xilinx Application Notes, XAPP739, XAPP740, and XAPP742.

SECTION IV: APPENDICES

HBlank and VBlank Periods for Standard Frames

Figure A-1 shows one Full video frame, an Active window, and horizontal and vertical blanking of a 1080p standard frame. Table A-1 shows Horizontal blank and Vertical blank periods calculation for standard frame formats. This helps you align the fsync period for various frame formats.

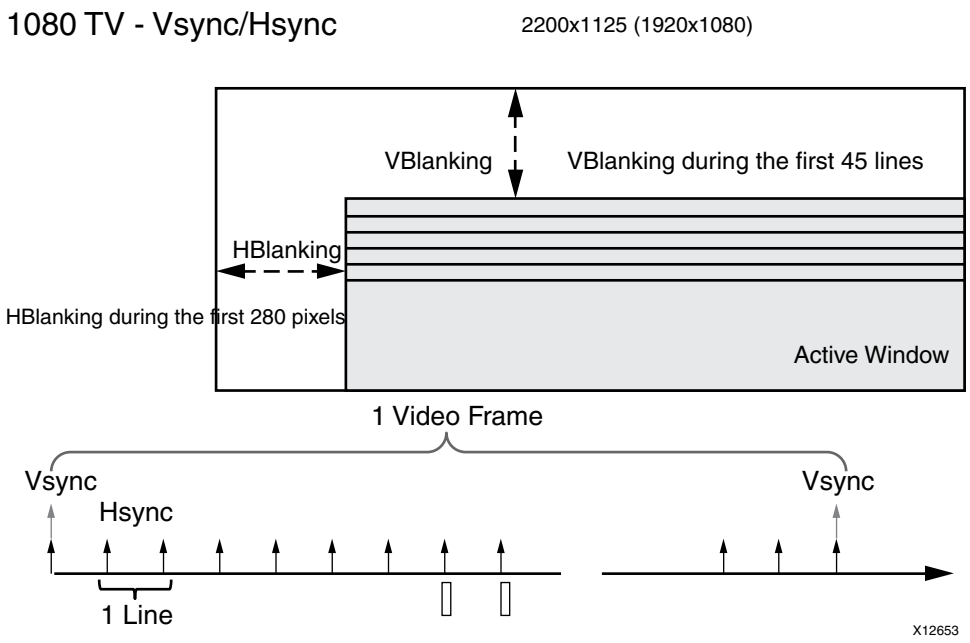


Figure A-1: **Frame Format of 1080p**

Table A-1: **HBlank and VBlank of Various Standard Frames**

	Video Display Standard	Total video format		Active video format		Nominal frame rate (Hz)	FSync or VSync period = 1/Nominal frame rate	HSync Period = Fsync or Vsync period/Total number of video lines	Vertical Blank or VBlank = (Total -Active) Video Lines*Hsync Period	Pixel Period = Hsync Period/Total Number of pixels	Horizontal Blank or HBlank = (Total -Active) Pixels*Pixel Period
		Number of pixels	Number of Lines	Number of pixels	Number of Lines						
PAL interlaced	576i	864	625	720	576	25	40 ms	64 µs	49*Hsync Period	0.074 µs	144*Pixel Period
PAL progressive	576p	864	625	720	576	50	20 ms	32 µs	49*Hsync Period	0.037 µs	144*Pixel Period
NTSC interlaced	480i	858	525	720	480	30	33.33 ms	63.49 µs	45*Hsync Period	0.074 µs	138*Pixel Period
NTSC progressive	480p	858	525	720	480	60	16.66 ms	31.74 µs	45*Hsync Period	0.037 µs	138*Pixel Period
HD1080i	1080i	2200	1125	1920	1080	30	33.33 ms	26.63 µs	45*Hsync Period	0.0134 µs	200*Pixel Period
HD1080p	1080p	2200	1125	1920	1080	60	16.66 ms	14.81 µs	45*Hsync Period	0.0067 µs	200*Pixel Period

Appendix B

Migrating

Special Considerations when Migrating to AXI

No action is required when upgrading from v4_00_a, v5_00_a and v5_01_a to v5_02_a. However, when upgrading from v3_00_a to v5_02_a, in EDK, remove the following VDMA parameters from the mhs file.

- c_s_axi_lite_aclk_freq_hz
- c_m_axi_mm2s_aclk_freq_hz
- c_m_axi_s2mm_aclk_freq_hz
- c_m_axi_sg_aclk_freq_hz

When upgrading from v3_00_a to v5_02_a, manually connect streaming resets out provided by MM2S and S2MM channels if & as required (`mm2s_prmry_reset_out_n`, `s2mm_prmry_reset_out_n`).

Note: In v3_00_a these reset outputs were part of AXI-S streaming bus signals.

For Vivado™ and CORE™ Generator tools, there is an **UPGRADE IP** option available in the GUI that you can click to switch from v3_00_a, v4_00_a, v_5_00_a and v5_01_a to the latest version v5_02_a.

See *Vivado Design Suite Migration Methodology Guide*. (UG911) for more information about migration.

Debugging

Here are common issues that user encounters:

- AXI VDMA works, but bottom few lines are not proper
- AXI VDMA locks up
- Frame pointers does not work

To start with, most of these issues are attributed to wrong hsize and vsize programming in AXI VDMA. Double check programmed values and check if fsync signals are connected and fsync period is maintained properly.

Also, it is recommended to have C_MM2S_SOF_ENABLE = 1, C_S2MM_SOF_ENABLE = 1, C_USE_FSYNC = 1 and C_FLUSH_ON_FSYNC = 1 where AXI VDMA continues to work irrespective of data path errors. Setting C_S2MM_SOF_ENABLE = 1 also provides access to fsync cross-bar (FsyncSrcSelect: (bits[6:5] of S2MM_DMACR - offset 0x30h) through which different fsync sources can be selected.

The following registers are implemented to help debug the failures:

- Bits 7(SOFEarlyErr), 8(EOLEarlyErr), 11(SOFLateErr) and 15(EOLLateErr) of offset 0x34h to identify error occurrence in S2MM path
- S2MM hsize status register (bits [15:0] of offset 0xF0h) captures incoming hsize count during EOLEarly error
- S2MM vsize status register (bits [12:0] of offset 0xF4h) captures incoming vsize count during SOFEarly error
- FRMPTR_STS register (offset 0x24h) captures current operating frame pointer status of both MM2S and S2MM channels.

When everything fails, try applying soft reset (bit[2] of offset 0x00h for MM2S and bit[2] of offset 0x30h for S2MM) and re-initialize.

See Xilinx Solution Centers for information helpful to the debugging progress.

Appendix D

Additional Resources

Xilinx Resources

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

www.xilinx.com/support.

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

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

For a comprehensive listing of Video and Imaging application notes, white papers, reference designs and related IP cores, see the Video and Imaging Resources page at:

www.xilinx.com/esp/video/refdes_listing.htm

Solution Centers

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

References

These documents provide supplemental material useful with this user guide:

- *AXI Interconnect IP Data Sheet* (DS768)
- *AXI Reference Guide* (UG761)
- *AMBA AXI4-Stream Protocol Specification*
- Vivado Design Suite - 2012.2 User Guides web page.

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

Technical Support

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

Ordering Information

This Xilinx® LogiCORE IP module is provided at no additional cost with the standard Xilinx ISE® Design Suite and ISE Embedded Edition (EDK) software 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 software, contact your local Xilinx sales representative.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
10/19/11	1.0	Initial Xilinx release.
01/18/12	1.1	Summary of Major Core Changes Added 32 Frame Stores support Added Internal Genlock support Added Frame Sync on TUSER0 support Added additional stream data width support Summary of Major Documentation Changes Removed List of Acronym from Appendix. For the first occurrence of each acronym, spelled out full text. Added supported software drivers to IP Facts table. Created new section Scatter Gather Mode. Reordered the hierarchy of the Register Space section. Reordered the hierarchy of the Designing with the Core section. Added the new section, Triple Frame Buffer Example. Added new Appendix, HBlank and VBlank Periods for Standard Frames
04/24/12	1.2	Summary of Major Core Changes Added independent fsync control for both MM2S and S2MM channels Added Dynamic Genlock support
07/25/12	1.3	Updated to core version 5.02.a and 14.2 ISE tools. Added Vivado tools and Zynq™-7000 support. Updated Error section. Updated many items in the IP Facts table. Added Additional Design section.

Notice of Disclaimer