AXI DMA v7.1

LogiCORE IP Product Guide

Vivado Design Suite

PG021 June 14, 2019

目录

知识产权事实

- 简介

  Xilinx 逻辑内核 AXI DMA(AXI Direct Memory)是为了使用 Xilinx Vivado 设计工具而设计的软核。在内存和 AXI 流目标外设之间提供了高带宽的内存直连访问。

- 特性

1. AXI4 兼容
2. 支持可供选择的加扰/解扰内存直连访问
3. AXI4 数据宽度支持：32、64、128、256、512、1024bits
4. AXI4 流模式支持数据位宽：8、16、32、64、128、256、512、1024bits
5. 可选锁孔支架
6. 可供选择的数据重新对其支持流数据宽度高达 512bits
7. 支持 AXI 控制和状态流
8. 支持可选择的多 DMA
9. 支持高达 64bit 地址

概述

AXI DMA IP 核在 AXI4 内存映射和 AXI4-流传输接口之间提供了高带宽内存直连访问。在基于处理器的系统中，其可选的分散-收集功能还减轻了中央处理器（CPU）的数据移动任务。通过 AXI4-Lite 从机接口可访问初始化、状态和管理寄存器。表 1-1 描述了内核的功能组成。

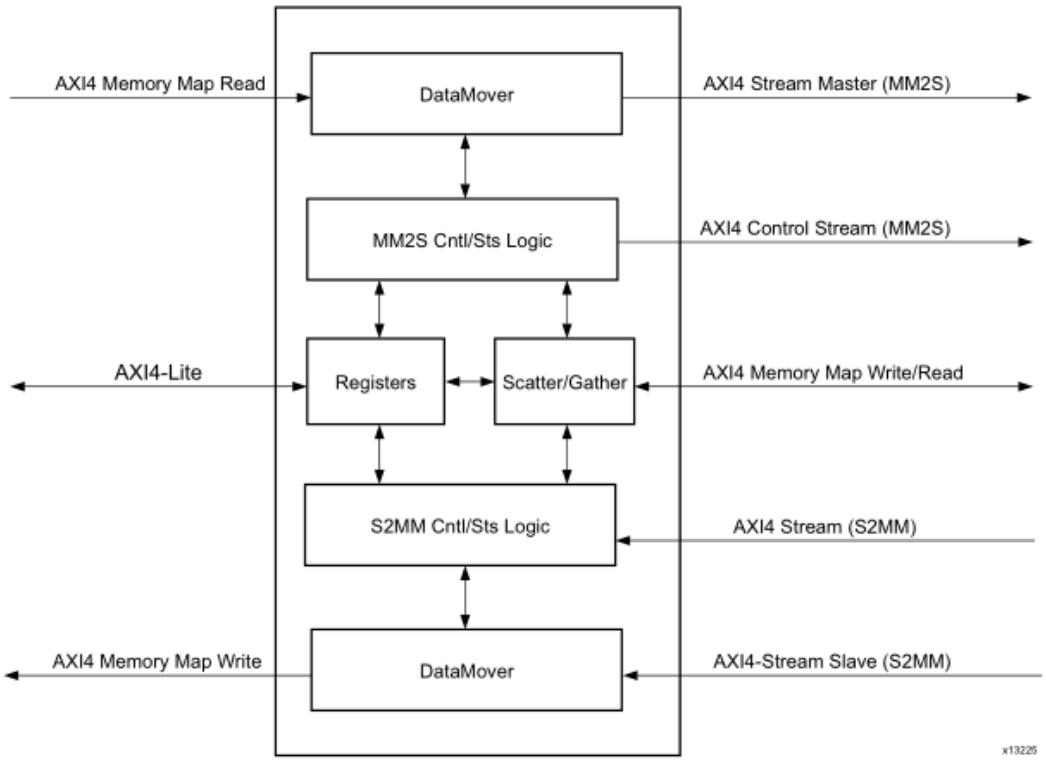

Figure 1-1: **AXI DMA Block Diagram**

系统内存和流目标之间的主要高速 DMA 数据移动是通过 AXI4 读主设备到 AXI4 内存映射到流（MM2S）主设备，以及 AXI 流到内存映射（S2MM）从映射到 AXI4 写主设备。 AXI DMA 还可以在分散/聚集模式下在 MM2S 和 S2MM 路径上最多支持 16 个多通道的数据移动。

MM2S 通道和 S2MM 通道独立运行。 AXI DMA 提供 4 KB 地址边界保护（在非 Micro DMA 中配置时），自动突发映射，并提供使用 AXI4-Stream 总线的几乎全带宽功能对多个传输请求进行排队的功能。 此外，AXI DMA 提供字节级数据重新对齐，从而允许在任何字节偏移量位置开始进行内存读写。

MM2S 通道支持 AXI 控制流，用于将用户应用程序数据发送到目标 IP。 对于 S2MM 通道，提供了 AXI 状态流，用于从目标 IP 接收用户应用程序数据。

可选的 Scatter / Gather 引擎通过 AXI4 Scatter Gather 读/写主接口从系统内存中获取并更新缓冲区描述符。

功能摘要

1. 符合 AXI4
2. 可选的独立分散/聚集直接内存访问（DMA）支持
   a) 提供从 CPU 卸载 DMA 管理工作的功能
   b) 提供与主要数据总线无关的获取和更新传输描述符的功能
   c) 允许将描述符放置在任何内存中映射的位置与数据缓冲区分开。 例如，描述符可以放置在 Block RAM 中。
   d) 提供可选的循环操作
3. 可选的直接寄存器模式（不支持分散/收集）通过排除分散收集引擎，可以启用性能较低但 FPGA 资源占用较少的模式。 在这种模式下，通过设置源地址（对于 MM2S）或目标地址（对于 S2MM），然后在长度寄存器中指定字节数来命令传输。
4. 支持 32、64、128、256、512 和 1,024 位的主要 AXI4 数据宽度
5. 支持 8、16、32、64、128、256、512 和 1,024 位的主要 AXI4-Stream 数据宽度
6. 可选的数据重新对齐 流数据宽度最大为 512 位的引擎。允许将数据重新对齐到主内存映射和流数据路径上的字节（8 位）级别
7. 可选的 AXI 控制和状态流，可与 AXI 以太网 IP 接口，为 MM2S 通道提供可选的控制流，为 S2MM 通道提供低状态的状态流 -高带宽数据路径中的-带宽控制和状态。
8. 可选的 Micro 模式。AXI DMA 可以配置为提供低占用空间，低性能的 IP，可以处理小数据包的传输。 阅读以下章节以获取更多信息。

应用领域

AXI DMA 提供了系统内存与基于 AXI4-Stream 的目标 IP（例如 AXI 以太网）之间的高速数据移动。

许可和订购

根据 Xilinx 最终用户许可的条款，XilinxVivado® 设计套件将免费提供此 Xilinx®LogiCORE™IP 模块。

有关此模块和其他 Xilinx LogiCORE IP 模块的信息，请参见 Xilinx 知识产权页面。 有关其他 Xilinx LogiCORE IP 模块和工具的价格和可用性的信息，请联系当地的 Xilinx 销售代表。

产品规格书

性能

本节包含以下小节。

- 最大频率
- 延迟和吞吐量

最大频率

AXI DMA 按照《 Vivado 设计套件用户指南：使用 IP 设计（UG896）[参考 1]》中描述的基准测试方法进行表征。 表 2-1 显示了表征运行的结果。

表 2-1：最大频率

Table 2-1: **Maximum Frequencies**

Family	Speed Grade	Fmax (MHz)		
		AXI4	AXI4-Stream	AXI4-Lite
Virtex®-7	–1	200	200	180
Kintex®-7		200	200	180
Artix®-7		150	150	120
Virtex-7	–2	240	240	200
Kintex-7		240	240	200
Artix-7		180	180	140
Virtex-7	–3	280	280	220
Kintex-7		280	280	220
Artix-7		200	200	160

延迟和吞吐量

表 2-2 和表 2-3 描述了 AXI DMA 的等待时间和吞吐量。 这些表提供了典型配置的性能信息。 吞吐量测试包括在 MM2S 和 S2MM 侧传输 10,000 个字节。

从完成描述符获取（DMACR.Idle = 1）到帧计数中断断言，一直测量吞吐量。

表 2-2：AXI DMA 延迟数量

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

表 2-3：AXI DMA 吞吐量数字（1）

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

注意：
1.上图是使用默认 IP 配置测得的。
2. MM2S 吞吐量是在“内存映射”端的第一个有效端与“流”端的“ tlast”之间测量的。
3. S2MM 吞吐量是在流媒体端的第一个 tvalid 到内存映射端的最后一个 wlast 之间进行测量的

资源利用率

有关性能和资源利用率的完整详细信息，请访问“性能和资源利用率”网页。

端口说明

表 2-4 中描述了 AXI DMA I / O 信号。

信号名称	接口	信号类型	初始状态	描述
s_axi_lite_aclk	Clock	I		AXI4-Lite 时钟。
m_axi_sg_aclk	Clock	I		AXI DMA 分散收集时钟
m_axi_mm2s_aclk	Clock	I		AXI DMA MM2S 主时钟
m_axi_s2mm_aclk	Clock	I		AXI DMA S2MM 主时钟
axi_resetn	Reset	I		AXI DMA 复位。 低电平有效复位。 置为低电平时，将重置整个 AXI DMA 内核。 必须与 s_axi_lite_aclk 同步。
mm2s_introut	Interrupt	O	0	中断输出以将内存映射到流通道。
s2mm_introut	Interrupt	O	0	中断输出以流到内存映射通道。
axi_dma_tstvec	NA	O	0	调试信号供内部使用。
AXI4-Lite 接口信号				
s_axi_lite_*	S_AXI_LITE	Input/ Output		有关 AXI4 信号，请参见 AXI 参考指南（UG1037）的附录 A [Ref 2]。
MM2S 内存映射读取接口信号				
m_axi_mm2s_*	M_AXI_MM2S	Input/ Output		有关 AXI4 信号，请参见 AAXI 参考指南（UG1037）[Ref 2]的附录 A。
MM2S 主流接口信号				
mm2s_prmry_reset_out_n	M_AXIS_MM2S	O	1	主 MM2S 复位输出。 低电平有效复位。
m_axis_mm2s_*	M_AXIS_MM2S	Input/ Output		有关 AXI4 信号，请参见 AXI 参考指南（UG1037）的附录 A [Ref 2]。
MM2S 主控流接口信号				
mm2s_cntrl_reset_out_n	M_AXIS_CNTRL	O		控制重置。 低电平有效复位。
m_axis_mm2s_cntrl_*	M_AXIS_CNTRL	Input/ Output	1	有关 AXI4 信号，请参见 AXI 参考指南（UG1037）的附录 A [Ref 2]。
m_axi_s2mm_*	M_AXI_S2MM	Input/ Output		有关 AXI4 信号，请参见 AXI 参考指南（UG1037）的附录 A [Ref 2]。
S2MM 从站流接口信号				
s2mm_prmry_reset_out_n	S_AXIS_S2MM	O	1	主 S2MM 复位输出。 低电平有效复位。
s_axis_s2mm_*	S_AXIS_S2MM	I	Input/ Output	有关 AXI4 信号，请参见 AXI 参考指南（UG1037）的附录 A [Ref 2]。
S2MM 从站状态流接口信号				
s2mm_sts_reset_out_n	S_AXIS_STS	O	1	AXI 状态流（STS）复位输出。 低电平有效复位。
s_axis_s2mm_sts_*	S_AXIS_STS	Input/ Output		有关 AXI4 信号，请参见 AXI 参考指南（UG1037）的附录 A [Ref 2]。
分散收集内存映射读取接口信号				
m_axi_sg_*	M_AXI_SG	Input/ Output		有关 AXI4 信号，请参见 AXI 参考指南（UG1037）的附录 A [Ref 2]。
分散收集内存映射写入接口信号				
m_axi_sg*	M_AXI_SG	Input/ Output		有关 AXI4 信号，请参见 AXI 参考指南（UG1037）的附录 A [Ref 2]。

寄存器空间

表 2-5 中显示了分散/聚集模式的 AXI DMA 内核寄存器空间。 表 2-6 中显示了用于直接寄存器模式的 AXI DMA 内核寄存器空间。 AXI DMA 寄存器被内存映射到不可缓存的内存空间。 该存储空间必须在 AXI 字（32 位）边界上对齐。

注意：AXI4-Lite 写访问寄存器由 32 位 AXI 写数据（*_wdata）信号更新，并且不受 AXI 写数据选通（*_wstrb）信号的影响。 对于写操作，应同时声明 AXI 写地址有效（*_awvalid）和 AXI 写数据有效（*_wvalid）信号。

Endianess

所有寄存器均为 Little Endian 格式，如图 2-1 所示。

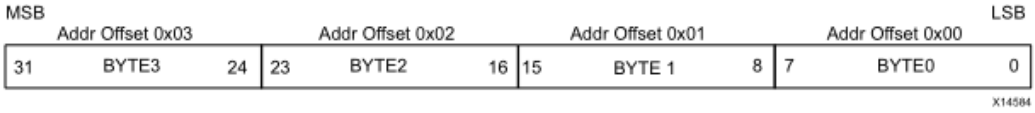

Figure 2-1: **32-bit Little Endian Example**

AXI DMA 寄存器地址映射

表 2-5：分散/收集模式寄存器地址映射

地址偏移	名称	描述
00h	MM2S_DMACR	MM2S DMA 控制寄存器
04h	MM2S_DMASR	MM2S DMA 状态寄存器
08h	MM2S_CURDESC	MM2S 当前描述符指针。 地址的低 32 位。
0Ch	MM2S_CURDESC_MSB	MM2S 当前描述符指针。 地址的高 32 位。
10h	MM2S_TAILDESC	MM2S 尾部描述符指针。 低 32 位。
14h	MM2S_TAILDESC_MSB	MM2S 尾部描述符指针。 地址的高 32 位。
2Ch	SG_CTL	分散/收集用户和缓存
30h	S2MM_DMACR	S2MM DMA 控制寄存器
34h	S2MM_DMASR	S2MM DMA 状态寄存器
38h	S2MM_CURDESC	S2MM 当前描述符指针。 低 32 位地址
3Ch	S2MM_CURDESC_MSB	S2MM 当前描述符指针。 高 32 位地址。
40h	S2MM_TAILDESC	S2MM 尾部描述符指针。 低 32 位地址。
44h	S2MM_TAILDESC_MSB	S2MM 尾部描述符指针。 高 32 位地址。

注意：

1.地址空间偏移与 C_BASEADDR 分配有关。

2.仅当在多通道模式下配置 DMA 时，寄存器 2Ch 才可用。

表 2-6：直接寄存器模式寄存器地址映射

地址偏移	名称	描述
00h	MM2S_DMACR	MM2S DMA 控制寄存器
04h	MM2S_DMASR	MM2S DMA 状态寄存器
08-14h	Reserved	N/A
18h	MM2S_SA	MM2S 源地址。 地址的低 32 位。
1Ch	MM2S_SA_MSB	MM2S 源地址。 地址的高 32 位。
28h	MM2S_LENGTH	MM2S 传输长度（字节）
30h	S2MM_DMACR	S2MM DMA 控制寄存器
34h	S2MM_DMASR	S2MM DMA 状态寄存器
38-44h	Reserved	N/A
48h	S2MM_DA	S2MM 目标地址。 低 32 位地址。
4Ch	S2MM_DA_MSB	S2MM 目标地址。 高 32 位地址。
58h	S2MM_LENGTH	S2MM 缓冲区长度（字节）

注意：
1.地址空间偏移与 C_BASEADDR 分配有关。

存储器映射到流寄存器详细信息

寄存器访问类型说明

•RO =只读。 写入无效
•R / W =读写可访问
•R / WC =读写清除

MM2S_DMACR（MM2S DMA 控制寄存器–偏移量 00h）

该寄存器提供对内存映射到流 DMA 通道的控制。

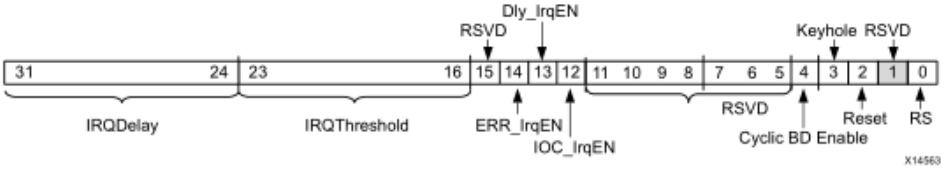

Figure 2-2: **MM2S DMACR Register**

表 2-7：MM2S_DMACR 寄存器的详细信息

bits	field Name	默认值	Access Type	描述
0	RS	0	R/W	运行/停止控制，用于控制 DMA 通道的运行和停止。 •0 =停止–当前（如果有）DMA 操作完成时，DMA 停止。 对于分散/聚集模式，待处理的命令/传输被刷新或完成。 AXI4-Stream 输出可能会提前终止。 在引擎停止之前，允许更新队列中的描述符完成对远程内存的更新。 对于直接寄存器模式，未完成的命令/传输被刷新或完成。 AXI4-Stream 输出可能会终止。 当 DMA 引擎停止时，DMA 状态寄存器中的停止位置为 1。 发生错误时，AXI DMA 硬件将清除该位。 CPU 也可以选择清除该位以停止 DMA 操作。 •1 =运行–启动 DMA 操作。 当 DMA 引擎开始运行时，DMA 状态寄存器中的中止位置为 0。
1	Reserved	1	RO	写入该位无效，并且始终读为 1。
2	Reset	0	R/W	软复位，用于复位 AXI DMA 内核。 将此位设置为 1 将导致 AXI DMA 复位。 重置可以正常完成。 待处理的命令/传输已刷新或完成。 AXI4-Stream 输出可能会提前终止。 设置 MM2S_DMACR.Reset = 1 或 S2MM_DMACR.Reset = 1 将重置整个 AXI DMA 引擎。 软复位完成后，所有寄存器和位均处于复位状态。 •0 =正常运行。 •1 =正在进行重置。
3	Keyhole	1	R/W	锁孔读取。 将此位设置为 1 会导致 AXI DMA 在非递增地址模式（AXI4 上的固定地址突发传输）中启动 MM2S 读取（AXI4read）。 当 AXI DMA 空闲时，可以更新此位。 使用锁孔操作时，最大突发长度不应超过 16。 启用 DRE 时，不应设置此位。 当启用多通道功能或处于直接寄存器模式时，该位不起作用。
4	Cyclic BD Enable	0	R/W	设置为 1 时，DMA 在循环缓冲区描述符（BD）模式下运行，而无需任何用户干预。 在这种模式下，分散收集模块将忽略 BD 的"已完成"位。 设置此位后，您可以循环使用相同的 BD，而不必担心任何过时的描述符错误。 仅当 DMA 空闲或不运行时，才应设置/取消设置该位。 在 DMA 运行时更新此位可能会导致意外行为。 当 DMA 在多通道模式下运行时，该位不起作用。
11 to 5	Reserved	0	R/O	写入这些位无效，并且它们始终读为零。
12	IOC_IrqEn	0	R/W	完成中断（IOC）中断使能。 设置为 1 时，允许 DMASR.IOC_Irq 为设置了 IOC 位的描述符生成中断输出。 •0 =禁止 IOC 中断 •1 =允许 IOC 中断
13	Dly_IrqEn	0	R/W	延迟定时器中断使能中断。 设置为 1 时，允许 DMASR.Dly_Irq 产生中断输出。 •0 =禁止延迟中断 •1 =使能延迟中断 注：当 AXI DMA 配置为直接寄存器模式时，该位将被忽略。
14	Err_IrqEn	0	R/W	错误时中断中断使能。 •0 =禁止错误中断 •1 =允许错误中断
15	Reserved	0	R/O	写入该位无效，并且始终读为零。
23 to 16	IRQThreshold	01H	R/W	中断阈值。 该值用于设置中断阈值。 当发生 IOC 中断事件时，内部计数器将从"中断阈值"设置中递减计数。 当计数达到零时，DMA 引擎将产生中断输出。 注意：阈值的最小设置为 0x01。 向该寄存器写入 0x00 无效。 注意：当将 AXI DMA 配置为直接寄存器模式时，将忽略此字段。
31 to 24	IRQDelay	00h	R/W	中断延迟超时。 该值用于设置中断超时值。 延迟超时机制到期后，中断超时机制使 DMA 引擎生成中断。 计时器在数据包末尾开始计数，并在收到新数据包或发生超时事件时复位。 1 Timeout Interval 125 clock period of SG clock 此处将值设置为 3 会导致 125 x 3 x（SG 时钟的时钟周期）的延迟超时。 注意：将此值设置为零将禁用延迟计时器中断。 注意：当将 AXI DMA 配置为直接寄存器模式时，将忽略此字段。

MM2S_DMASR (MM2S DMA Status Register – Offset 04h)

该寄存器提供了内存映射到流 DMA 通道的状态。

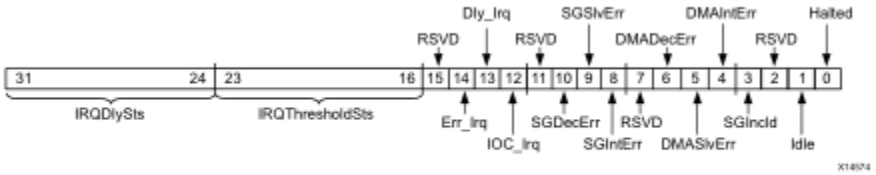

Figure 2-3: **MM2S DMASR Register**

表 2-8：MM2S_DMASR 寄存器的详细信息

bits	field Name	默认值	Access Type	描述
0	Halted	1	RO	DMA 通道已暂停。 指示 DMA 通道的运行/停止状态。 •0 = DMA 通道正在运行。 •1 = DMA 通道停止。 对于分散/收集模式，当 DMACR.RS = 0 并且 DMA 和分散收集（SG）操作已停止时，该位置 1。 对于直接寄存器模式（C_INCLUDE_SG = 0），当 DMACR.RS = 0 并且 DMA 操作停止时，该位置 1。 从 DMACR.RS = 0 到 DMASR.Halted = 1 之间可能会有时间间隔。 注：暂停（RS = 0 和暂停= 1）时，在分散收集模式下，写入 TAILDESC_PTR 指针寄存器对 DMA 操作无效。 对于直接寄存器模式，写 LENGTH 寄存器对 DMA 操作没有影响。
1	Idle	0	RO	DMA 通道空闲。 指示 AXI DMA 操作的状态。 对于分散/聚集模式，当 IDLE 指示 SG 引擎已到达关联通道的尾指针且所有排队的描述符已处理时。 写入尾指针寄存器会自动重新启动 DMA 操作。 空闲位与 BD 相关联。 DMA 可能处于 IDLE 状态，AXI 接口上可能有活动数据。 对于直接寄存器模式，当 IDLE 指示当前传输已完成时。 •0 =不空闲。 对于分散/聚集模式，SG 尚未到达尾描述符描述符指针和/或正在进行的 DMA 操作。对于直接寄存器模式，传输未完成。 •1 =空闲。 对于分散/聚集模式，SG 已到达尾部描述符指针，并且 DMA 操作已暂停。 对于直接寄存器模式，DMA 传输已完成并且控制器已暂停。 注：通道暂停（DMASR.Halted = 1）时，该位为 0。 当 AXI DMA 配置为直接寄存器模式时，该位在初始传输之前也为 0。
2	Reserved	0	RO	写入该位无效，并且始终读为零。
3	SGIncld	0	RO	1 =启用分散收集 0 =未启用分散收集
4	DMAIntErr	0	RO	DMA 内部错误。 如果在提取的描述符中指定的缓冲区长度设置为 0，则会发生内部错误。此错误情况导致 AXI DMA 正常停止。 DMACR.RS 位设置为 0，并且当引擎完全关闭时，DMASR.Halted 位设置为 1。 •0 =没有 DMA 内部错误 •1 =检测到 DMA 内部错误。 DMA 引擎停止。 注意：当 AXI DMA 配置为直接寄存器模式时，该位不使用，固定为 0。
5	DMASlvErr	0	RO	DMA 从站错误。 如果从内存映射接口读取的从设备发出从设备错误，则会发生此错误。 此错误情况导致 AXI DMA 正常停止。 DMACR.RS 位设置为 0，并且当引擎完全关闭时，DMASR.Halted 位设置为 1。 •0 =无 DMA 从错误。 •1 =检测到 DMA 从错误。 DMA 引擎停止。
6	DMADecErr	0	RO	DMA 解码错误。 如果地址请求指向无效的地址，则会发生此错误。此错误情况导致 AXI DMA 正常停止。 DMACR.RS 位设置为 0，并且当引擎完全关闭时，DMASR.Halted 位设置为 1。 •0 =无 DMA 解码错误。

				•1 =检测到 DMA 解码错误。 DMA 引擎停止。
7	Reserved	0	RO	写入该位无效，并且始终读为零。
8	SGIntErr	0	RO	散点图内部错误。 如果获取已设置"完成位"的描述符，则会发生此错误。 有关更多信息，请参阅"分散收集描述符"部分。这向 SG 引擎表明该描述符是陈旧的描述符。 此错误情况导致 AXI DMA 正常停止。 DMACR.RS 位设置为 0，并且当引擎完全关闭时，DMASR.Halted 位设置为 1。 •0 =无 SG 内部错误。 •1 =检测到 SG 内部错误。 DMA 引擎停止。 注意：当 AXI DMA 配置为直接寄存器模式时，该位不使用，固定为 0。
9	SGSlvErr	0	RO	散点收集从站错误。 如果从"内存映射"接口上读取的从设备发出从设备错误，则会发生此错误。 此错误情况导致 AXI DMA 正常停止。 DMACR.RS 位设置为 0，并且当引擎完全关闭时，DMASR.Halted 位设置为 1。 •0 =无 SG 从站错误。 •1 =检测到 SG 从站错误。 DMA 引擎停止。 注意：当 AXI DMA 配置为直接寄存器模式时，该位不使用，固定为 0。
10	SGDecErr	0	RO	分散收集解码错误。 如果 CURDESC_PTR 和/或 NXTDESC_PTR 指向无效的地址，则会发生此错误。 此错误情况导致 AXI DMA 正常停止。 DMACR.RS 位设置为 0，并且当引擎完全关闭时，DMASR.Halted 位设置为 1。 •0 =无 SG 解码错误。 •1 =检测到 SG 解码错误。 DMA 引擎停止。 注意：当 AXI DMA 配置为直接寄存器模式时，该位不使用，固定为 0。
11	Reserved	0	RO	写入该位无效，并且始终读为零。
12	IOC_Irq	0	R/WC	完成时中断。 当分散/聚集模式设置为 1 时，表示在描述符完成时产生了中断事件。 对于设置了帧结束（EOF）位的描述符，会发生这种情况。 当直接寄存器模式设置为 1 时，表示在传输完成时产生了中断事件。 如果 MM2S_DMACR 中的相应位被使能（IOC_IrqEn = 1），并且如果满足中断阈值，则会导致从 AXI DMA 产生中断。 •0 =无 IOC 中断。 •1 =检测到 IOC 中断。 向该位写入 1 将清除它。
13	Dly_Irq	0	R/WC	延迟中断。 设置为 1 时，表示在延迟定时器超时时产生了中断事件。 如果 MM2S_DMACR 中的相应位使能（Dly_IrqEn = 1），则会从 AXI DMA 产生中断输出。 •0 =无延迟中断。 •1 =检测到延迟中断。1=检测到 IOC 中断。 向该位写入 1 将清除它。 注意：当 AXI DMA 配置为直接寄存器模式时，该位不使用，固定为 0。
14	Err_Irq	0	R/WC	•错误中断。 设置为 1 时，表示由于错误而产生了中断事件。 如果在 MM2S_DMACR 中使能了相应的位（Err_IrqEn = 1），则会从 AXI DMA 产生中断输出。 向该位写入 1 将清除它。 •0 =无错误中断。 •1 =检测到错误中断
15	Reserved	0	RO	始终读为零。
23-16	IRQThresholdSts	01h	RO	中断阈值状态。 指示当前中断阈值。 MM2S_CR 的 IRQThreshold 字段中编程的值在每次传输数据包时都会递减，并在此处反映出来。 在启动 DMA 之前或发送第一个数据包之前，此寄存器的值将与 MM2S_CR 的 IRQThreshold 字段中编程的值相同。 注意：仅在启用“散布收集”时适用。
31-24	IRQDelaySts	00h	RO	中断延迟时间状态。 指示当前中断延迟时间值。 注意：仅在启用“散布收集”时适用。

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

该寄存器为存储器映射提供了当前描述符指针，以进行流 DMA 分散聚集描述符管理。

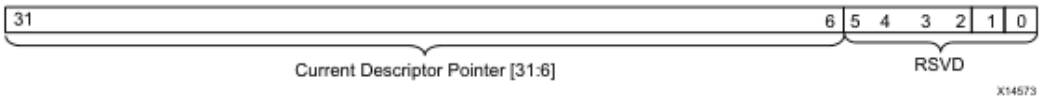

Figure 2-4: **MM2S CURDESC Register**

bits	field Name	默认值	Access Type	描述
5 to 0 (Offset 0x38)	Reserved	0	RO	写入这些位无效，它们始终读为零。
31 to 6	Current Descriptor Pointer	zeros	R/W(RO)	指示正在处理的当前描述符的指针。 在写 TAILDESC_PTR 寄存器之前，该寄存器必须包含一个指向有效描述符的指针。 否则，将产生不确定的结果。 当 DMACR.RS 为 1 时，CURDESC_PTR 变为只读（RO），并用于获取第一个描述符。 当 DMA 引擎正在运行(DMACR.RS = 1)时，AXI DMA 更新 CURDESC_PTR 寄存器以指示正在使用的当前描述符。 错误检测时，将更新 CURDESC_PTR 以反映与检测到的错误关联的描述符 注意：仅当 DMA 引擎暂停时（DMACR.RS = 0 和 DMASR.Halted = 1），该寄存器才可以由 CPU 写入。 在其他所有时间，该寄存器均为只读（RO）。 描述符必须对齐 16 个字，即 0x00、0x40、0x80 等。 任何其他比对都有不确定的结果。

MM2S_CURDES_MSB (MM2S DMA Current Descriptor Pointer Register -

Offset 0Ch)

该寄存器为存储器映射提供了当前描述符指针的高 32 位，以进行流 DMA 分散聚集描述符管理。 仅当地址空间大于 32 位时才适用。

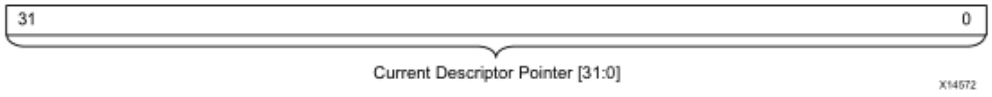

Figure 2-5: **MM2S CURDESC_MSB Register**

表 2-10：MM2S_CURDESC_MSB 寄存器详细信息

bits	field Name	默认值	Access Type	描述
31-0	当前描述符指针	zeros	R/W (RO)	指示正在处理的当前描述符的指针。 在写 TAILDESC_PTR 寄存器之前，该寄存器必须包含一个指向有效描述符的指针。 否则，将产生不确定的结果。 当 DMACR.RS 为 1 时，CURDESC_PTR 变为只读（RO），并用于获取第一个描述符。 当 DMA 引擎正在运行(DMACR.RS = 1)时，AXI DMA 更新 CURDESC_PTR 寄存器以指示正在使用的当前描述符。 错误检测时，将更新 CURDESC_PTR 以反映与检测到的错误关联的描述符。 注意：仅当 DMA 引擎暂停时（DMACR.RS = 0 和 DMASR.Halted = 1），该寄存器才可以由 CPU 写入。 在其他所有时间，该寄存器均为只读（RO）。 描述符必须对齐 16 个字，即 0x00、0x40、0x80 等。 任何其他比对都有不确定的结果。

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

该寄存器为存储器映射提供尾部描述符指针，以流 DMA 散点图收集器描述符管理。

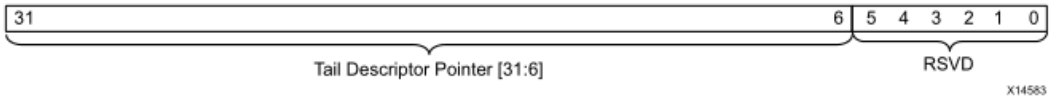

Figure 2-6: **MM2S_TAILDESC Register**

表 2-11：MM2S_TAILDESC 寄存器的详细信息

bits	field Name	默认值	Access Type	描述
5-0	保留	0	RO	写入这些位无效，它们总是读为零。
31 to 6	尾描述符指针	0	R/W	指示描述符链中的暂停指针。在对当前描述符指针与尾部描述符指针匹配的描述符完成操作之后，AXI DMA SG 引擎会暂停描述符获取。 当未停止 AXI DMA 通道（DMASR.Halted = 0）时，CPU 对 TAILDESC_PTR 寄存器的写操作将导致 AXI DMA SG 引擎开始获取描述符，或者在空闲时重新启动描述符（DMASR.Idle = 1）。 如果它不是空闲的，则写 TAILDESC_PTR 除了重新定位暂停点没有任何作用。 注意：软件不得将尾指针移至尚未更新的位置。 软件处理并重新分配所有完成的描述符（Cmplted = 1），清除完成的位，然后移动尾指针。 软件必须将指针移到它更新的最后一个描述符。 描述符必须以 16 个字对齐，即 0x00、0x40、0x80，依此类推。 任何其他比对都有不确定的结果。

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

该寄存器提供尾部描述符指针的高 32 位用于存储器映射，以流 DMA 散点图收集器描述符管理。 仅当地址空间超过 32 位宽时才适用。

Figure 2-7: **MM2S_TAILDESC_MSB Register**

表 2-12：MM2S_TAILDESC_MSB 寄存器详细信息

bits	field Name	默认值	Access Type	描述
31 to 0	尾描述符指针	0	R/W	指示描述符链中的暂停指针。 在对当前描述符指针与尾部描述符指针匹配的描述符完成操作之后，AXI DMA SG 引擎会暂停描述符获取。 当未停止 AXI DMA 通道（DMASR.Halted = 0）时，CPU 对 TAILDESC_PTR_MSB 寄存器的写入将导致 AXI DMA SG 引擎开始获取描述符，或者在空闲时重新启动描述符（DMASR.Idle = 1）。 如果它不是空闲的，则写 TAILDESC_PTR 除了重新定位暂停点没有任何作用。 如果 AXI DMA 通道被暂停（DMASR.Halted = 1 且 DMACR.RS = 0），则 CPU 对 TAILDESC_PTR 寄存器的写操作仅对重新定位暂停点有效。 注意：软件不得将尾指针移至尚未更新的位置。 软件处理并重新分配所有完成的描述符（Cmplted = 1），清除完成的位，然后移动尾指针。 软件必须将指针移到它更新的最后一个描述符。 描述符必须以 16 个字对齐，即 0x00、0x40、0x80，依此类推。 任何其他比对都有不确定的结果。

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

该寄存器提供了用于读取系统内存的源地址，以进行内存映射到流 DMA 的传输。

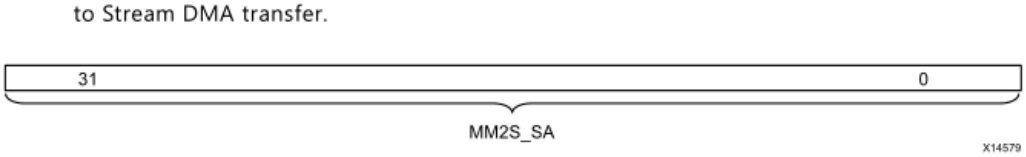

Figure 2-8: **MM2S_SA Register**

bits	field Name	默认值	Access Type	描述
31 to 0	源地址	zeros	R/W	指示从 AXI DMA 读取的源地址，以将数据传输到 MM2S 通道上的 AXI4-Stream。 注意：如果包括数据重新对齐引擎，则源地址可以处于任何字节偏移处。如果不包括数据重新对齐引擎，则源地址必须与 MM2S 内存映射数据宽度对齐。

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

该寄存器提供源地址的高 32 位，用于读取系统内存以进行“存储器映射到流 DMA”传输。 这仅在 DMA 配置为大于 32 的地址空间时适用。

Figure 2-9: **MM2S_SA_MSB Register**

bits	field Name	默认值	Access Type	描述
31 to 0	源地址	zeros	R/W	指示从源地址 AXI DMA 读取的 MSB 32 位，以将数据传输到 MM2S 通道上的 AXI4-Stream。 注意：如果包括数据重新对齐引擎，则源地址可以在任何字节偏移处。如果不包括数据重新对齐引擎，则源地址必须与 MM2S 内存映射数据宽度对齐。

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

该寄存器提供了从系统内存读取并传输到 MM2S AXI4-Stream 的字节数。

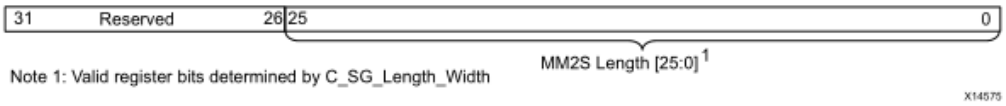

Figure 2-10: **MM2S_LENGTH Register**

bits	field Name	默认值	Access Type	描述
25 (1) to 0	长度	zeros	R/W	指示要为 MM2S 通道传输的字节数。 向该寄存器写入非零值将启动 MM2S 传输。
31 to 26	Reserved	0	RO	写入这些位无效，它们始终读为零。

注意：
1.“长度的宽度”字段由缓冲区长度寄存器的宽度参数确定。 最小宽度为 8 位（7 到 0），最大宽度为 26 位（25 到 0）。

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

仅当 DMA 配置为多通道模式时，此寄存器才可用。

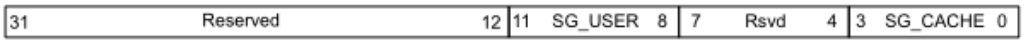

Figure 2-11: **SG_CTL Register**

表 2-16：SG_CTL 寄存器详细信息

bits	field Name	默认值	Access Type	描述
3-0	SG_CACHE	0011b	R/W	分散/聚集缓存控制。 写入该寄存器的值反映在 M_AXI_SG 接口的 m_axi_sg_arcache 和 m_axi_sg_awcache 信号上。
7-4	Reserved	0	RO	写入这些位无效，它们始终读为零。
11-8	SG_USER	0	R/W	散布/收集用户控件。 写入该寄存器的值反映在 M_AXI_SG 接口的 m_axi_sg_aruser 和 m_axi_sg_awuser 信号上。

流到内存映射寄存器详细信息

S2MM_DMACR (S2MM DMA Control Register – Offset 30h)

该寄存器提供对流到存储器映射 DMA 通道的控制。

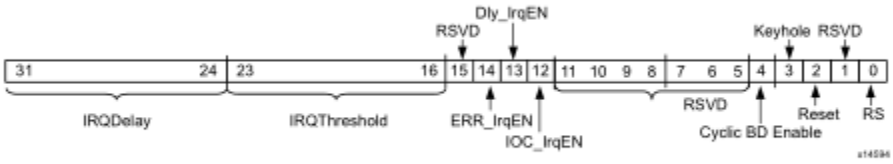

Figure 2-12: **S2MM DMACR Register**

表 2-17：S2MM_DMACR 寄存器的详细信息

bits	field Name	默认值	Access Type	描述
0	RS	0	R/W	运行/停止控制，用于控制 DMA 通道的运行和停止。 •0 =停止–当前（如果有）DMA 操作完成时，DMA 停止。 对于分散/聚集模式，待处理的命令/传输被刷新或完成。 AXI4-Streams 可能提前终止。在引擎停止之前，允许更新队列中的描述符完成对远程内存的更新。 对于直接寄存器模式，未完成的命令/传输被刷新或完成。 AXI4-Streams 可能会终止。 不能保证 S2MM AXI4 上的数据完整性。 当 DMA 引擎停止时，DMA 状态寄存器中的停止位置为 1。 发生错误时，AXI DMA 硬件将清除该位。 CPU 也可以选择清除该位以停止 DMA 操作。 •1 =运行–启动 DMA 操作。 当 DMA 引擎开始运行时，DMA 状态寄存器中的中止位置为 0。
1	RESERVED	1	RO	写入该位无效，并且始终读为 1。
2	Reset	0	R/W	软复位，用于复位 AXI DMA 内核。 将此位设置为 1 将导致 AXI DMA 复位。 重置可以正常完成。 待处理的命令/传输已刷新或完成。 如有必要，可通过关联的 TLAST 提前终止 AXI4-Stream 输出。 设置 MM2S_DMACR.Reset = 1 或 S2MM_DMACR.Reset = 1 将重置整个 AXI DMA 引擎。 软复位完成后，所有寄存器和位均处于复位状态。 •0 =重置未进行。 普通手术。 •1 =正在进行重置。
3	Keyhole	0	R/W	锁孔入路。 将此位设置为 1 会导致 AXI DMA 在非递增地址模式（AXI4 上的固定地址突发传输）中启动 S2MM 写（AXI4 写）。 当 AXI DMA 空闲时，可以修改此位。 启用锁眼操作时，最大突发长度不能超过 16。启用 DRE 时，不应设置此位。 当在多通道模式下使用 DMA 时，该位不起作用。
4	Cyclic BD Enable	0	R/W	设置为 1 时，您可以在循环缓冲区描述符（BD）模式下使用 DMA，而无需任何用户干预。 在这种模式下，分散收集模块将忽略 BD 的"已完成"位。 使用此功能，您可以循环使用相同的 BD，而不必担心任何过时的描述符错误。 当 DMA 在多通道模式或直接寄存器模式下运行时，该位不起作用。
11-5	Reserved	0	RO	写入这些位无效，它们始终读为零。
12	IOC_IRqEn	0	R/W	完全中断使能时中断。 设置为 1 时，允许"完成时中断"事件为设置了"完成位"位的描述符生成中断输出。 •0 =禁止 IOC 中断。 •1 =使能 IOC 中断。
13	Dly_IrqEn	0	R/W	延迟定时器中断使能中断。 设置为 1 时，允许错误事件产生中断。 0 =禁止延迟中断。

				•1 =使能延迟中断。 注意：仅在启用“散布收集”时适用。
14	Err_IrqEn	0	R/W	错误时中断中断使能。 设置为 1 时，允许错误事件产生中断。 •0 =禁止错误中断。 •1 =使能错误中断。
15	Reserved	0	RO	写入该位无效，并且始终读为零。
23 to 16	IRQThreshold	01h	R/W	中断阈值。 该值用于设置中断阈值。 当发生 IOC 中断事件时，内部计数器将从“中断阈值”设置中递减计数。 当计数达到零时，DMA 引擎将产生中断输出。 注意：阈值的最小设置为 0x01。 向该寄存器写入 0x00 无效。 注意：仅在启用“散布收集”时适用。
31 to 24	IRQDelay	00h	R/W	中断延迟超时。 该值用于设置中断超时值。 中断超时是一种机制，用于使 DMA 引擎在延迟时间段到期后生成中断。 计时器在数据包末尾开始计数，并在收到新数据包或发生超时事件时复位。 1 Timeout Interval = 125 *(clock period of SG clock) 此处将值设置为 3 会导致 125 x 3 x（SG 时钟的时钟周期）的延迟超时。 注意：将此值设置为零将禁用延迟计时器中断。 注意：仅在启用“散布收集”时适用。

S2MM_DMASR (S2MM DMA Status Register – Offset 34h)

该寄存器提供流到内存映射 DMA 通道的状态。

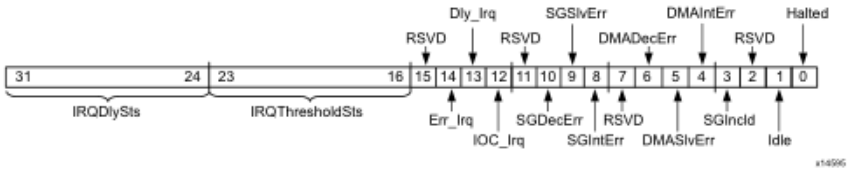

Figure 2-13: **S2MM DMASR Register**

表 2-18：S2MM_DMASR 寄存器的详细信息

bits	field Name	默认值	Access Type	描述
0	Halted	1	RO	DMA 通道已暂停。 指示 DMA 通道的运行/停止状态。 •0 = DMA 通道正在运行。 •1 = DMA 通道停止。对于分散/聚集模式，当 DMACR.RS = 0 且 DMA 和 SG 操作停止时，该位置 1。 对于直接寄存器模式，当 DMACR.RS = 0 并且 DMA 操作停止时，该位置 1。 从 DMACR.RS = 0 到 DMASR.Halted = 1 之间可能会有时间间隔。 注：暂停（RS = 0 和暂停= 1）时，在分散收集模式下，写入 TAILDESC_PTR 指针寄存器对 DMA 操作无效。对于直接寄存器模式，写 LENGTH 寄存器对 DMA 操作没有影响。
1	Idle	0	RO	DMA 通道空闲。 指示 AXI DMA 操作的状态。 对于分散/聚集模式，当 IDLE 指示 SG 引擎已到达关联通道的尾指针且所有排队的描述符已处理时。 写入尾指针寄存器会自动重新启动 DMA 操作。 对于直接寄存器模式，当 IDLE 指示当前传输已完成时。 •0 =不空闲。 •1 =空闲。 注：通道暂停（DMASR.Halted = 1）时，该位为 0。 当 AXI DMA 配置为直接寄存器模式时，该位在初始传输之前也为 0。
2	Reserved	0	RO	写入该位无效，并且始终读为零。
3	SGIncld	C_ INCLUDE_ SG	RO	包括分散收集引擎。 DMASR.SGIncld = 1 表示已包含 Scatter Gather 引擎，并且 AXI DMA 配置为 Scatter Gather 模式。 DMASR.SGIncld = 0 表示排除了分散收集引擎，并且 AXI DMA 配置为直接寄存器模式。
4	DMAIntErr	0	RO	DMA 内部错误。如果在提取的描述符中指定的缓冲区长度设置为 0，则会发生此错误。此外，在分散收集模式下并使用 status app length 字段时，当 Status AXI4-Stream 数据包 RxLength 字段与 S2MM 数据包不匹配时，会发生此错误。 由 S_AXIS_S2MM 接口接收。 禁用“分散收集”时，如果在存储器写入过程中发生任何错误，或者如果传入的数据包大于 DMA 长度寄存器中指定的数据包，则会标记此错误。 此错误情况导致 AXI DMA 正常停止。 DMACR.RS 位设置为 0，并且当引擎完全关闭时，DMASR.Halted 位设置为 1。 •0 =无 DMA 内部错误。 •1 =检测到 DMA 内部错误。
5	DMASlvErr	0	RO	DMA 从站错误。如果从内存映射接口读取的从设备发出从设备错误，则会发生此错误。此错误情况导致 AXI DMA 正常停止。 DMACR.RS 位设置为 0，并且当引擎完全关闭 DMASR.Halted 位设置为 1 时。 •0 =无 DMA 从错误。 •1 =检测到 DMA 从错误。
6	DMADecErr	0	RO	DMA 解码错误。 如果地址请求指向无效的地址，则会发生此错误。此错误情况导致 AXI DMA 正常停止。 DMACR.RS 位设置为 0，并且当引擎完全关闭时，DMASR.Halted 位设置为 1。 •0 =无 DMA 解码错误。 •1 =检测到 DMA 解码错误。
7	Reserved	0	RO	写入该位无效，并且始终读为零。
8	SGIntErr	0	RO	散点图内部错误。 如果获取已设置了 Complete 位的描述符，则会发生此错误。 这向 SG 引擎指示该描述符是尾部描述符。 此错误情况导致 AXI DMA 正常停止。 DMACR.RS 位设置为 0，并且当引擎完全关闭时，DMASR.Halted 位设置为 1。 •0 =无 SG 内部错误。 •1 =检测到 SG 内部错误。 该错误无法登录到描述符中。 注意：仅在启用“散布收集”时适用
9	SGSlvErr	0	RO	散点收集从站错误。 如果从“内存映射”接口上读取的从设备发出“从设备错误”，则会发生此错误。 此错误情况导致 AXI DMA 正常停止。DMACR.RS 位设置为 0，并且当引擎完全关闭时，DMASR.Halted 位设置为 1。 •0 =无 SG 从站错误。 •1 =检测到 SG 从站错误。 DMA 引擎停止。

				该错误无法登录到描述符中。 注意：仅在启用“散布收集”时适用。
10	SGDecErr	0	RO	分散收集解码错误。 如果 CURDESC_PTR 和/或 NXTDESC_PTR 指向无效的地址，则会发生此错误。 此错误情况导致 AXI DMA 正常停止。 DMACR.RS 位设置为 0，并且当引擎完全关闭时，DMASR.Halted 位设置为 1。 •0 =无 SG 解码错误。 •1 =检测到 SG 解码错误。 DMA 引擎停止。 该错误无法登录到描述符中。 注意：仅在启用“散布收集”时适用。
11	Reserved	0	RO	写入该位无效，并且始终读为零。
12	IOC_Irq	0	R/WC	完成时中断。 当散射/聚集模式设置为 1 时，表明在描述符完成时产生了中断事件。 对于设置了帧结束（EOF）位的描述符，会发生这种情况。 当直接寄存器模式设置为 1 时，表示在传输完成时产生了中断事件。 如果使能了 S2MM_DMACR 中的相应位（IOC_IrqEn = 1），并且如果满足中断阈值，则会导致从 AXI DMA 产生中断。 •0 =无 IOC 中断。 •1 =检测到 IOC 中断。 向该位写入 1 将清除它。
13	Dly_Irq	0	R/WC	延迟中断。 设置为 1 时，表示在延迟定时器超时时产生了中断事件。 如果使能了 S2MM_DMACR 中的相应位（Dly_IrqEn = 1），则会从 AXI DMA 产生中断输出。 •0 =无延迟中断。 •1 =检测到延迟中断。 向该位写入 1 将清除它。 注意：仅在启用“散布收集”时适用。
14	Err_Irq	0	R/WC	错误中断。 设置为 1 时，表示由于错误而产生了中断事件。 如果使能了 S2MM_DMACR 中的相应位（Err_IrqEn = 1），则会从 AXI DMA 产生中断输出。 向该位写入 1 将清除它。 •0 =无错误中断。 •1 =检测到错误中断。
15	Reserved	0	RO	写入该位无效，并且始终读为零。
23-16	IRQThresholdSts	01	RO	中断阈值状态。 指示当前中断阈值。 在每次数据包传输时，在 S2MM_CR 的 IRQThreshold 字段中编程的值都会减小，并在此处反映出来。 在启动 DMA 之前或在接收第一个数据包之前，此寄存器的值将与 S2MM_CR 的 IRQThreshold 字段中编程的值相同。 注意：仅在启用“散布收集”时适用。
31-24	IRQDelaySts	00	RO	中断延迟时间状态。 指示当前中断延迟时间值。 注意：仅在启用“散布收集”时适用。

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

该寄存器为流到内存映射 DMA 分散收集描述符管理提供了当前描述符指针。

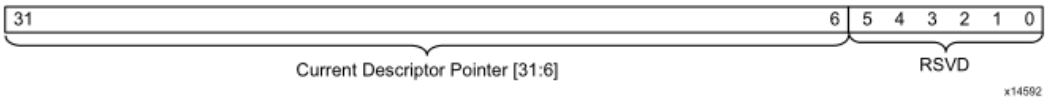

Figure 2-14: **S2MM CURDESC Register**

表 2-19：S2MM_CURDESC 寄存器详细信息

bits	field Name	默认值	Access Type	描述
5 to 0 (Offset 0x38)	Reserved	0	RO	写入这些位无效，它们始终读为零。
31 to 6	当前描述符指针	0	R/W (RO)	指示正在处理的当前缓冲区描述符的指针。 在写入 TAILDESC_PTR 寄存器之前，该寄存器必须包含指向有效描述符的指针。 否则，将产生不确定的结果。 当 DMACR.RS 为 1 时，CURDESC_PTR 变为只读（RO），并用于获取第一个描述符。 当 DMA 引擎正在运行(DMACR.RS = 1)时，AXI DMA 更新 CURDESC_PTR 寄存器以指示正在使用的当前描述符。 错误检测时，将更新 CURDESC_PTR 以反映与检测到的错误关联的描述符。 注意：仅当 DMA 引擎停止时（DMACR.RS = 0 和 DMASR.Halted = 1），该寄存器才可以由 CPU 写入。 在其他所有时间，该寄存器均为只读（RO）。 缓冲区描述符必须是 16 个字对齐的，即 0x00、0x40、0x80 等。 任何其他比对都有不确定的结果。

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

该寄存器提供了当前描述符指针的高 32 位，用于流到内存映射 DMA 分散聚集描述符管理。仅在将 DMA 配置为大于 32 位的地址空间时使用。

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

bits	field Name	默认值	Access Type	描述
31 to 0	当前描述符指针	0	R/W (RO)	指示正在处理的当前缓冲区描述符的指针。 在写入 TAILDESC_PTR 寄存器之前，该寄存器必须包含一个指向有效描述符的指针。 否则，将产生不确定的结果。 当 DMACR.RS 为 1 时，CURDESC_PTR 变为只读（RO），并用于获取第一个描述符。 当 DMA 引擎正在运行（DMACR.RS = 1）时，AXI DMA 会更新 CURDESC_PTR 寄存器，以指示正在使用的当前描述符。 检测错误时，将更新 CURDESC_PTR 以反映与检测到的错误关联的描述符。 注意：仅当 DMA 引擎停止时（DMACR.RS = 0 和 DMASR.Halted = 1），该寄存器才可以由 CPU 写入。 在其他所有时间，该寄存器均为只读（RO）。

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

该寄存器为流到内存映射 DMA 分散收集描述符管理器提供了尾描述符指针。

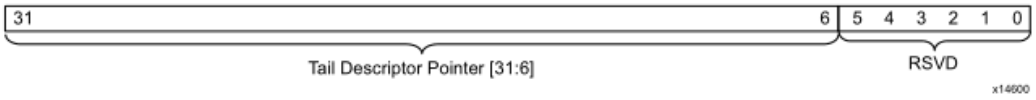

Figure 2-16: **S2MM TAILDESC Register**

表 2-21：S2MM_TAILDESC 寄存器详细信息

bits	field Name	默认值	Access Type	描述
5 to 0 (Offset 0x38)	Reserved	0	RO	写入这些位无效，它们始终读为零。
31 to 6	尾描述符指针	0	R/W	指示描述符链中的暂停指针。在对当前描述符指针与尾部描述符指针匹配的描述符执行完操作之后，AXI DMA SG 引擎会暂停描述符获取。 当未停止 AXI DMA 通道（DMASR.Halted = 0）时，CPU 对 TAILDESC_PTR 寄存器的写操作将导致 AXI DMA SG 引擎开始获取描述符，或者在空闲时重新启动描述符（DMASR.Idle = 1）。 如果它不是空闲的，则对 TAILDESC_PTR 的写入仅对重新定位暂停点无效。 如果将 AXI DMA 通道 DMACR.RS 位设置为 0（DMASR.Halted = 1 且 DMACR.RS = 0），则 CPU 对 TAILDESC_PTR 寄存器的写操作只会重新放置暂停点，否则无效。 注意：软件不得将尾巴指针移动到尚未更新的位置。 软件处理并重新分配所有完成的描述符（Cmplted = 1），清除完成的位，然后移动尾指针。软件必须将指针移动到它更新的最后一个描述符。 描述符必须以 16 个字对齐，即 0x00、0x40、0x80，依此类推。 任何其他比对都有不确定的结果。（RO）。 缓冲区描述符必须是 16 个字对齐的，即 0x00、0x40、0x80 等。 任何其他比对都有不确定的结果。

S2MM_TAILDESC_MSB (S2MM DMA Tail Descriptor Pointer Register – Offset44h)

该寄存器提供了尾部描述符指针的高 32 位，用于流到内存映射 DMA 分散聚集描述符管理。当 DMA 配置为大于 32 的地址空间时使用。

Figure 2-17: **S2MM TAILDESC_MSB Register**

表 2-22：S2MM_TAILDESC_MSB 寄存器详细信息

bits	field Name	默认值	Access Type	描述
31 to 0	尾描述符指针	zeros	R/W	指示描述符链中的暂停指针。在对当前描述符指针与尾部描述符指针匹配的描述符完成操作之后，AXI DMA SG 引擎会暂停描述符获取。 当未停止 AXI DMA 通道（DMASR.Halted = 0）时，CPU 对 TAILDESC_PTR_MSB 寄存器的写入将导致 AXI DMA SG 引擎开始获取描述符，或者在空闲时重新启动描述符（DMASR.Idle = 1）。 如果它不是空闲的，则对 TAILDESC_PTR 的写入仅对重新定位暂停点无效。 如果将 AXI DMA 通道 DMACR.RS 位设置为 0（DMASR.Halted = 1 且 DMACR.RS = 0），则 CPU 对 TAILDESC_PTR 寄存器的写操作将无效，除非重新放置暂停点。 注意：软件不得将尾巴指针移动到尚未更新的位置。 软件处理并重新分配所有完成的描述符（CmpIted = 1），清除完成的位，然后移动尾指针。软件必须将指针移到它更新的最后一个描述符。

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

该寄存器提供用于写入系统内存的目标地址，以实现流到内存映射到 DMA 的传输。

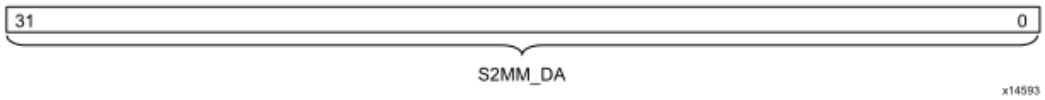

Figure 2-18: **S2MM_DA Register**

表 2-23：S2MM_DA 寄存器的详细信息

bits	field Name	默认值	Access Type	描述
31 to 0	目的地址	zeros	R/W	指示 AXI DMA 写入的目标地址，以从 S2MM 通道上的 AXI4-Stream 传输数据。 注意：如果包括数据重新对齐引擎，则目标地址可以是任意字节偏移量。如果不包括数据重新对齐引擎，则目标地址必须与 S2MM 内存映射数据宽度对齐。

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

该寄存器提供目标地址的高 32 位，用于写入系统存储器以实现流到存储器映射到 DMA 的传输。 仅在将 DMA 配置为大于 32 的地址空间时使用此选项。

Figure 2-19: **S2MM_DA_MSB Register**

bits	field Name	默认值	Access Type	描述
31 to 0	目的地址	zeros	R/W	指示目标地址的 MSB 32 位，AXI DMA 写入该目标地址以从 S2MM 通道上的 AXI4-Stream 传输数据。 注意：如果包括数据重新对齐引擎，则目标地址可以是任意字节偏移量。如果不包括数据重新对齐引擎，则目标地址必须与 S2MM 内存映射数据宽度对齐。

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

该寄存器提供缓冲区的长度（以字节为单位），以将数据从流写入内存映射 DMA 传输。

Figure 2-20: **S2MM_LENGTH Register**

表 2-25：S2MM_LENGTH 寄存器的详细信息

bits	field Name	默认值	Access Type	描述
25-0	长度	zeros	R/W	指示可用于写入来自 S2MM 通道的接收数据的 S2MM 缓冲区的字节长度。 向该寄存器写入非零值可使 S2MM 通道接收数据包数据。 S2MM 传输完成后，写入 S2MM AXI4 接口的实际字节数将更新到 S2MM_LENGTH 寄存器。 注意：该值必须大于或等于 S2MM AXI4-Stream 上要接收的最大预期数据包。 小于接收到的数据包的值会导致未定义的行为。
31 to 26	Reserved	0	RO	写入这些位无效，它们始终读为零。

注意：
1.“长度宽度”字段由“缓冲区长度寄存器宽度”参数确定。 最小宽度为 8 位（7 到 0），最大宽度为 26 位（25 到 0）。

散点图描述符

本节定义了将 AXI DMA 配置为分散/收集模式时 S2MM（接收）和 MM2S（发送）分散收集描述符的字段。 描述符由八个 32 位基本字和 0 或 5 个用户应用程序字组成。 该描述符将来支持 64 位地址，并支持用户应用程序数据。 通过“开始帧”和“结束帧”标志支持每个数据包多个描述符。

还包括完成状态和完成时中断。 每个描述符的缓冲区长度最多可以描述 8,388,607 字节的数据缓冲区。 MM2S 和 S2MM 这两个数据传输方向需要两个描述符链。

表 2-26：描述符字段（非多通道模式）

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

Notes:

注意：
1.地址空间偏移是相对于系统内存中的 16-32 位字对齐，即 0x00、0x40、0x80 等。

2.仅当包含控制/状态流时，才使用用户应用程序字段（APP0，APP1，APP2，APP3 和 APP4），当不包括控制/状态流时，用户应用程序字段不会被获取或更新。 散点图收集引擎。

3.仅当将 DMA 配置为大于 32 的地址空间时，才使用 MSB 字段或高 32 位地址。

MM2S_NXTDESC (MM2S Next Descriptor Pointer)

此值提供指向描述符链中下一个描述符的指针。

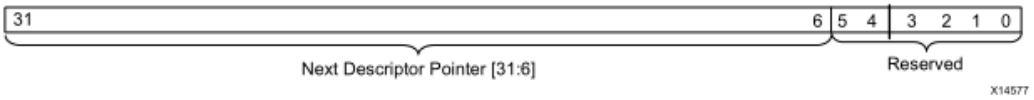

Figure 2-21: **MM2S_NXTDESC**

表 2-27：MM2S_NXTDESC 详细信息

bits	Field Name	描述
5 to 0	Reserved	这些位是保留位，应设置为零。
31 to 6	下一个描述符指针	指示指向下一个描述符的第一个单词的低阶指针。 注意：描述符必须以 16 个字对齐，即 0x00、0x40、0x80，依此类推。 任何其他比对都有不确定的结果。

MM2S_NXTDESC_MSB (MM2S Next Descriptor Pointer)

此值提供指向描述符链中下一个描述符的指针的高 32 位。仅当 AXI DMA 配置为大于 32 的地址空间时使用。

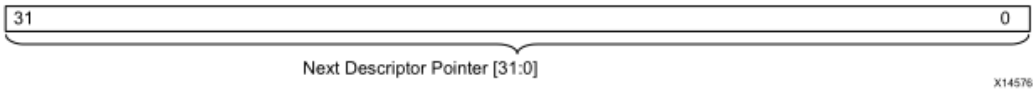

Figure 2-22: **MM2S_NXTDESC_MSB**

表 2-28：MM2S_NXTDESC_MSB 详细信息

bits	Field Name	描述
31 to 0	下一个描述符指针	指示指向下一个描述符的第一个字的指针的 MSB 32 位。

MM2S_BUFFER_ADDRESS (MM2S Buffer Address)

此值提供指向要从系统内存传输到流的数据缓冲区的指针。

Figure 2-23: **MM2S Buffer Address**

Table 2-29: MM2S_BUFFER_ADDRESS Details

bits	Field Name	描述
31 to 0	缓冲区地址	提供要从“内存映射”传输到“流”的数据位置。 注意：如果包括数据重新对齐引擎，则缓冲区地址可以是任意字节偏移量，但是缓冲区内的数据必须是连续的。 如果不包括数据重新对齐引擎，则缓冲区地址必须与 MM2S 内存映射数据宽度对齐。

MM2S_BUFFER_ADDRESS_MSB (MM2S Buffer Address)

此值提供指向数据缓冲区的指针的高 32 位，以将其从系统内存传输到流。 仅当 AXI DMA 配置为大于 32 的地址空间时使用。

Figure 2-24: **MM2S Buffer Address**

bits	Field Name	描述
31 to 0	缓冲区地址	提供要从内存映射传输到流的数据位置的 MSB 32 位。

MM2S_CONTROL (MM2S Control)

此值可控制从内存映射到流的 MM2S 传输。

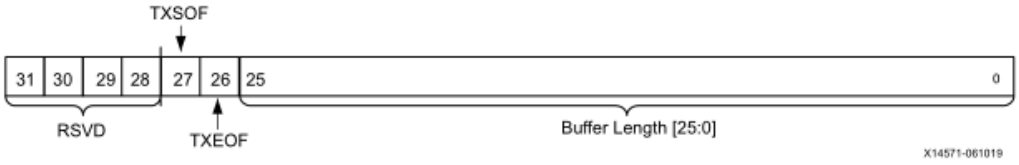

Figure 2-25: **MM2S_CONTROL**

表 2-31：MM2S_CONTROL 详细信息

bits	Field Name	描述
25-0	缓冲长度	指示传输缓冲区的大小（以字节为单位）。 该值指示要在 MM2S 流上发送的字节数。 缓冲区长度的可用宽度由参数"缓冲区长度寄存器的宽度"指定。 此字段最多可以描述 67,108,863 字节的传输。 在 Micro 模式下配置 AXI_DMA 时，该值不应超过以下公式：（MM2S 内存映射数据宽度/ 8）* Burst_length 注：将缓冲区长度寄存器宽度设置为小于 26 会降低 FPGA 资源利用率。
26	传输帧结束（TXEOF）	帧结束。 指示最后一个要处理的缓冲区的标志。 CPU 设置此标志，以向 AXI DMA 指示此描述符描述了数据包的结尾。 与此描述符关联的缓冲区最后发送。 •0 =非帧结束。 •1 =帧结束。 注意：为了正确操作，每个数据包必须有一个帧开始（SOF）描述符（TXSOF = 1）和一个帧结束（EOF）描述符（TXEOF = 1）。 使用单个描述符描述整个数据包是有效的，该描述符是 TXSOF = 1 和 TXEOF = 1 的描述符。
27	TXSOF	帧开始。 指示要处理的第一个缓冲区的标志。 CPU 设置此标志，以向 AXI DMA 指示此描述符描述了数据包的开始。 与此描述符关联的缓冲区首先被发送。 •0 =不开始帧。 •1 =帧开始。 注意：启用状态控制流后，从帧开始（SOF）描述符（TXSOF = 1）的 APP0 到 APP4 的用户应用程序数据将在控制流输出上传输。
31-28	Reserved	该位保留，应写为零。

MM2S_STATUS (MM2S Status)

此值提供 MM2S 从内存映射到流的传输状态。

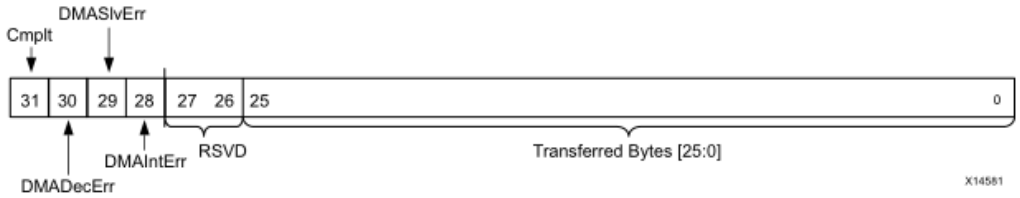

Figure 2-26: **MM2S_STATUS**

表 2-32：MM2S_STATUS 详细信息

bits	Field Name	描述
25-0	传输字节	指示为此描述符传输的实际数据的大小（以字节为单位）。 该值指示要在 MM2S 流上发送的字节数。 该值应与"控制缓冲区长度"字段匹配。 传输字节的可用宽度由缓冲区长度寄存器的宽度参数指定。 此字段最多可以描述 8388607 字节的传输。 在 Micro 模式下配置时，AXI_DMA 不会更新这些字段。 注意：将缓冲区长度寄存器宽度设置为小于 23 会降低 FPGA 资源利用率。 在 Micro 模式下配置 AXI_DMA 时，不会更新此字段。
27-26	Reserved	这些位是保留位，应设置为零。
28	DMAIntErr	DMA 内部错误。 主 AXI DataMover 检测到内部错误。 如果将要传输的长度为 0 的字节馈送到 AXI DataMover，则会发生此错误。 仅当在获取的描述符中指定的缓冲区长度设置为 0 时，才会发生这种情况。 此错误情况导致 AXI DMA 正常停止。 DMACR.RS 位设置为 0，并且当引擎完全关闭时，DMASR.Halted 位设置为 1。 •0 =无 DMA 内部错误。 •1 =检测到 DMA 内部错误。 DMA 引擎停止。
29	DMASlvErr	DMA 从站错误。 主 AXI DataMover 检测到从属错误。 如果从内存映射接口读取的从设备发出从设备错误，则会发生此错误。 此错误情况导致 AXI DMA 正常停止。 DMACR.RS 位设置为 0，并且当引擎完全关闭时，DMASR.Halted 位设置为 1。 •0 =无 DMA 从错误。 •1 =检测到 DMA 从错误。 DMA 引擎停止。
30	DMADecErr	DMA 解码错误。 主要 AXI DataMover 检测到解码错误。 如果描述符缓冲区地址指向无效地址，则会发生此错误。 此错误情况导致 AXI DMA 正常停止。 DMACR.RS 位设置为 0，并且当引擎完全关闭时，DMASR.Halted 位设置为 1。 •0 =无 DMA 解码错误。 •1 =检测到 DMA 解码错误。 DMA 引擎停止。
31	CmpIt	已完成。 这向软件指示 DMA 引擎已完成传输，如关联描述符所述。 传输完成后，DMA 引擎将此位设置为 1。 软件可以将 Completed 位设置为 1 来处理任何描述符。 •0 =描述符未完成。 •1 =描述符已完成。 注意：如果在此位设置为 1 的情况下获取描述符，则该描述符被视为过时的描述符。 标记了 SGIntErr，并且 AXI DMA 引擎停止。

MM2S_APP0 to MM2S_APP4(MM2S User Application Fields 0 to 4)

此值提供 MM2S 控制流的“用户应用程序”字段。

This value provides User Application fields for MM2S control stream.

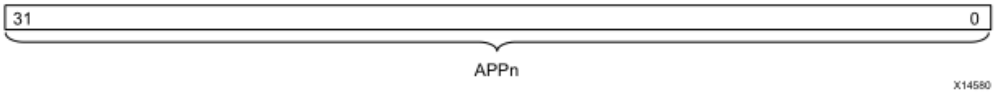

Figure 2-27: **MM2S_STATUS**

表 2-33：用户应用程序详细信息

bits	Field Name	描述
31 to 0	APP0 to APP4	用户应用程序字段 0 到 4。指定用户特定的应用程序数据。 启用状态控制流后，帧起始（SOF）描述符的应用程序（APP）字段将传输到 AXI 控制流。 对于 SOF = 0 的其他 MM2S 描述符，将提取 APP 字段，但将其忽略。 注意：如果未启用状态控制流，则不会获取这些字段。

S2MM_NXTDESC (S2MM Next Descriptor Pointer)

此值提供指向描述符链中下一个描述符的指针。

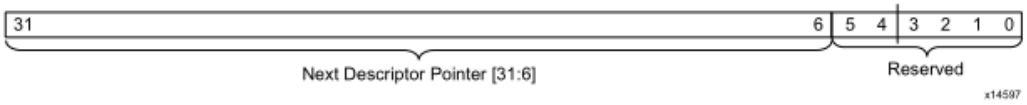

Figure 2-28: **S2MM_NXTDESC**

表 2-34：S2MM_NXTDESC 详细信息

bits	Field Name	描述
5 to 0	Reserved	这些位是保留位，应设置为零。
31 to 6	下一个描述符指针	指示指向下一个描述符的第一个单词的低阶指针。 注意：描述符必须以 16 个字对齐，即 0x00、0x40、0x80，依此类推。 任何其他比对都有不确定的结果。

S2MM_NXTDESC_MSB (S2MM Next Descriptor Pointer)

此值提供指向描述符链中下一个描述符的指针的高 32 位。仅当 AXI DMA 配置为大于 32 的地址空间时使用。

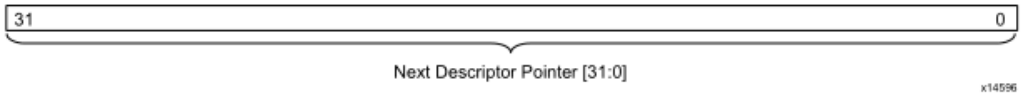

Figure 2-29: **S2MM_NXTDESC_MSB**

bits	Field Name	描述
31 to 0	下一个描述符指针	指示指向下一个描述符的第一个字的指针的 MSB 32 位。。

S2MM_BUFFER_ADDRESS (S2MM Buffer Address)

此值提供指向缓冲区的指针，该缓冲区可用于将数据从流传输到系统内存。

system memory.

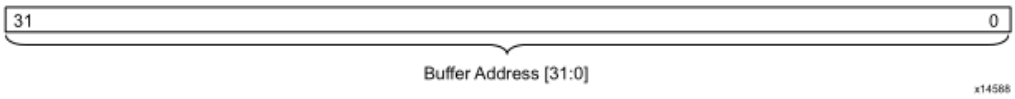

Figure 2-30: **S2MM Buffer Address**

表 2-36：S2MM_BUFFER_ADDRESS 详细信息

bits	Field Name	描述
31 to 0	缓冲区地址	提供可用于存储从流传输到内存映射的数据的缓冲区空间的位置。 注意：如果包括数据重新对齐引擎，则缓冲区地址可以在任何字节偏移处。 如果不包括数据重新对齐引擎，则缓冲区地址必须与 S2MM 内存映射的数据宽度对齐。

S2MM_BUFFER_ADDRESS_MSB (S2MM Buffer Address)

该值提供了指向缓冲区的指针的高 32 位，该缓冲区可用于将数据从流传输到系统内存。 仅当 AXI DMA 配置为大于 32 的地址空间时使用。

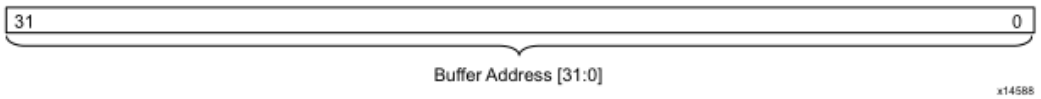

Figure 2-31: **S2MM Buffer Address (MSB)**

表 2-37：S2MM_BUFFER_ADDRESS_MSB 详细信息

bits	Field Name	描述
31 to 0	缓冲区地址	提供可用于存储从流传输到内存映射的数据的缓冲区空间位置的 MSB 32 位。

S2MM_CONTROL (S2MM Control)

此值可控制从流到内存映射的 S2MM 传输。

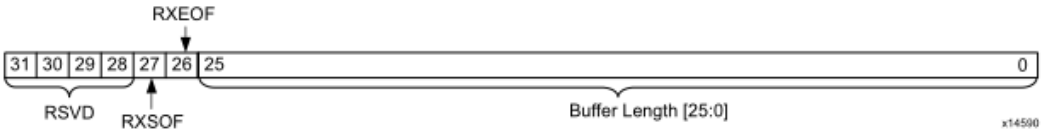

Figure 2-32: **S2MM_CONTROL**

表 2-38：S2MM_CONTROL 详细信息

bits	Field Name	描述
31 to 28	Reserved	这些位是保留位，应设置为零。
27	RXSOF	帧开始。 指示要处理的第一个缓冲区的标志。 sw / 用户设置此标志，以向 AXI DMA 指示此描述符描述了数据包的开始。 首先接收与此描述符关联的缓冲区。 •0 =非帧结束。 •1 =帧结束。 这仅适用于在 Micro 模式下配置 AXI_DMA 的情况。
26	接收帧结束	帧结束。 指示最后一个要处理的缓冲区的标志。 sw /用户设置此标志，以向 AXI DMA 指示此描述符描述了数据包的结尾。 与该描述符关联的缓冲区最后被接收。 •0 =非帧结束。 •1 =帧结束。 这仅适用于在 Micro 模式下配置 AXI_DMA 的情况。
25-0	缓冲长度	该值指示可用于接收 S2MM 流中的数据的空间（以字节为单位）。 缓冲区长度的可用宽度由参数"缓冲区长度寄存器的宽度"指定。 此字段最多可以描述 67,108,863 字节的传输。 注意：S2MM 描述符链中的总缓冲区空间（即，链中每个描述符的缓冲区长度值的总和）必须至少能够容纳最大接收数据包大小。 如果收到的数据包大于定义的缓冲区空间，则会发生不确定的结果。 注意：将缓冲区长度寄存器宽度设置为小于 23 会降低 FPGA 资源利用率。 注意：在 Micro 模式下配置 AXI_DMA 时，该值不应超过以下公式：（S2MM 内存映射数据宽度/ 8）* Burst_length

S2MM_STATUS (S2MM Status)

此值提供从流到内存映射的 S2MM 传输状态。

Figure 2-33: **S2MM_STATUS**

表 2-39：S2MM_STATUS 详细信息

bits	Field Name	描述
25-0	传输字节	此值指示此描述符描述的缓冲区中接收和存储的数据量。 这可能匹配缓冲区长度，也可能不匹配。 例如，如果此描述符指示的缓冲区长度为 1,024 字节，但仅接收到 50 个字节并将其存储在缓冲区中，则"传输的字节数"字段指示为 0x32。 可以通过将来自每个描述符的已传输字节值（来自 RXSOF 描述符）添加到接收帧结束（RXEOF）描述符中来确定整个接收包长度。 注意：传输字节的可用宽度由缓冲区长度寄存器的宽度参数指定。 此字段最多可以描述 67,108,863 字节的传输。 注意：将缓冲区长度寄存器宽度设置为小于 26 会降低 FPGA 资源利用率。在 Micro 模式下配置 AXI_DMA 时，不会更新此字段。
26	RXEOF	帧结束。 标志指示缓冲区保存数据包的最后部分。 该位由 AXI DMA 设置，以向 sw /用户指示与此描述符关联的缓冲区包含数据包的末尾。 •0 =非帧结束。 •1 =帧结束。 注意：启用控制/状态流后，通过状态流输入发送的用户应用程序数据将存储在 RXEOF 描述符的 APP0 至 APP4 中。
27	RXSOF	帧开始。 标志指示缓冲区保存数据包的第一部分。 该位由 AXI DMA 设置，以向 sw /用户指示与此描述符关联的缓冲区包含数据包的开始。 •0 =不开始帧。 •1 =帧开始。
28	DMAIntErr	DMA 内部错误。 主 AXI DataMover 检测到内部错误。如果将要传输的长度为 0 的字节馈送到 AXI DataMover，则会发生此错误。 仅当在获取的描述符中指定的"缓冲区长度"设置为 0 时，才会发生这种情况。如果发生欠载或超载情况，也可能导致此错误。 此错误情况导致 AXI DMA 正常停止。 DMACR.RS 位设置为 0，并且当引擎完全关闭时，DMASR.Halted 位设置为 1。 •0 =无 DMA 内部错误。 •1 =检测到 DMA 内部错误。 DMA 引擎停止。
29	DMASlvErr	DMA 从站错误。 主 AXI DataMover 检测到从属错误。如果从内存映射接口读取的从设备发出从设备错误，则会发生此错误。 此错误情况导致 AXI DMA 正常停止。 DMACR.RS 位设置为 0，并且当引擎完全关闭时，DMASR.Halted 位设置为 1。 •0 =无 DMA 从错误。 •1 =检测到 DMA 从错误。

		DMA 引擎停止。
30	DMADecErr	DMA 解码错误。 主要 AXI DataMover 检测到解码错误。 如果描述符缓冲区地址指向无效地址，则会发生此错误。 此错误情况导致 AXI DMA 正常停止。 DMACR.RS 位设置为 0，并且当引擎完全关闭时，DMASR.Halted 位设置为 1。 •0 =无 DMA 解码错误。 •1 =检测到 DMA 解码错误。 DMA 引擎停止。
31	CmpIt	已完成。 这向软件指示 DMA 引擎已完成传输，如关联描述符所述。 传输完成后，DMA 引擎将此位设置为 1。 该软件可以将 Completed 位设置为 1 来操纵任何描述符。 •0 =描述符未完成。 •1 =描述符已完成。 注意：如果在此位设置为 1 的情况下获取描述符，则该描述符被视为过时的描述符。一个 SGIntErr 被标记，AXI DMA 引擎停止。

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

该值为状态流上 S2MM 接收到的状态提供了“用户应用程序”字段空间。

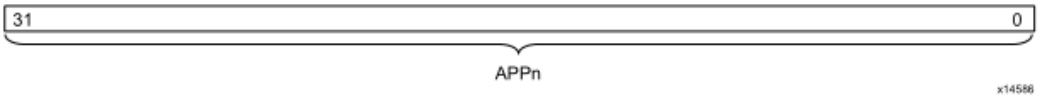

Figure 2-34: **S2MM_APP0 to S2MM_APP3**

表 2-40：用户应用程序 0 到 3 详细信息

bits	Field Name	描述
31 to 0	APP0 to APP3	启用状态/控制流后，在 AXI 状态流上接收的状态数据将存储到帧结束（EOF）描述符的 APP 字段中。 对于 EOF = 0 的其他 S2MM 描述符，Scatter Gather Engine 将 APP 字段设置为零。 注意：如果禁用状态/控制字段，则 Scatter Gather Engine 不会更新这些字段。

S2MM_APP4 (S2MM User Application Field 4)

该值为状态流上的 S2MM 接收状态提供了用户应用程序 4 字段空间。

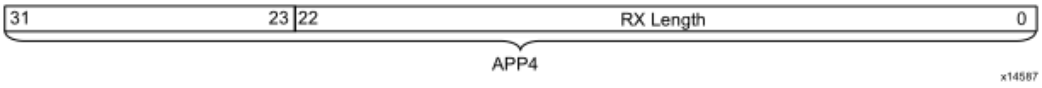

Figure 2-35: **S2MM_APP4**

表 2-41：用户应用程序 4 的详细信息

bits	Field Name	描述
31 to 0	APP4 / RxLength	用户应用程序字段 4 和接收字节长度。 如果未启用在状态流中使用 RxLength，则此字段的功能与 APP0 至 APP3 相同，因为在 AXI 状态流中接收到的状态数据存储在帧结束（EOF）描述符的 APP4 字段中。 启用“在状态流中使用 RxLength"时，此字段具有双重用途。 缓冲区长度寄存器宽度中指定的前最低有效位指定在 S2MM 主数据流上接收到的数据包的接收字节总数。 其次，其余的最高有效位是用户应用程序数据。

描述符管理

在开始 DMA 操作之前，软件应用程序必须建立描述符链。

当 AXI DMA 开始处理描述符时，它会获取，处理并更新描述符。 通过分析描述符，软件应用程序可以读取相关 DMA 传输的状态，在接收（S2MM）通道上获取用户信息，并确定传输完成。 利用此信息，软件应用程序可以管理描述符和数据缓冲区。

软件应用程序处理与完成的描述符关联的每个缓冲区，并重新分配描述符以供 AXI DMA 使用。 为防止软件和硬件相互踩踏，将创建尾指针模式。 尾指针由软件初始化，以指向描述符链的末尾。这成为硬件的暂停点。硬件开始运行时，它将获取并处理链中的每个描述符，直到到达尾指针为止。 然后，AXI DMA 暂停描述符处理。 允许软件处理和重新分配 Complete 位设置为 1 的任何描述符。

写入 TAILDESC 寄存器的操作会使 AXI DMA 硬件（如果在尾指针处暂停）再次开始处理描述符。 如果 AXI DMA 硬件没有在 TAILDESC 指针处暂停，则写入 TAILDESC 寄存器对硬件没有影响。 在这种情况下，AXI DMA 继续处理描述符，直到到达新的尾部描述符指针位置。描述符管理必须由软件完成。

AXI DMA 不管理描述符。

MM2S Descriptor Settings and AXI Control Stream

描述符 SOF / EOF 设置与 AXI 控制流之间的关系如图 2-36 所示。 SOF = 1 的描述符是数据包的开始，并为 MM2S 方向重置 DRE。 如果启用了状态/控制流，则此描述符的用户应用程序字段也会显示在 AXI 控制流上。 AXI DMA 引擎会忽略紧随 SOF = 1 的描述符之后的用户应用程序字段，直到并包括 EOF = 1 的描述符。 如果禁用了状态/控制流，则 SG Fetch Engine 不会提取 User Application 字段。

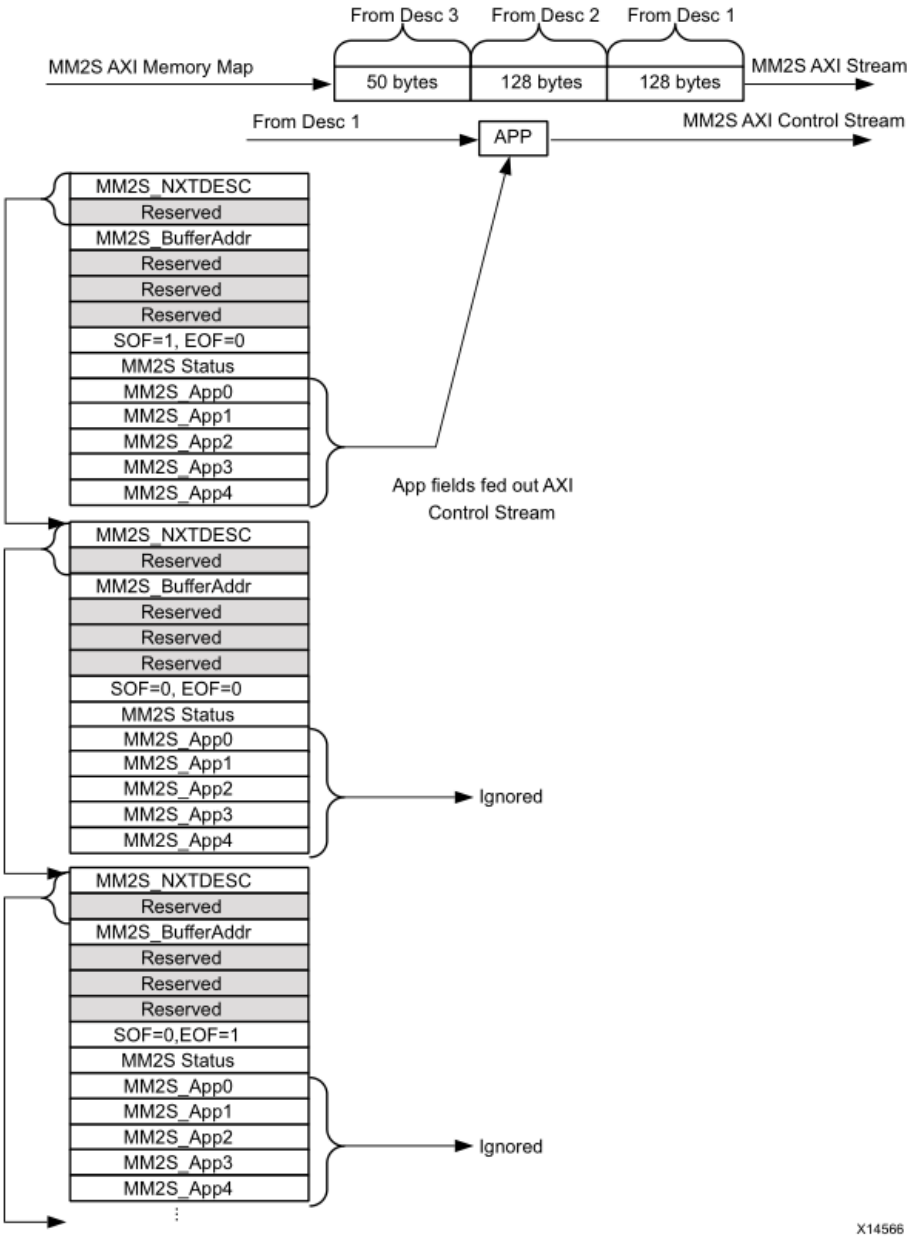

Figure 2-36: **Detail of Descriptor Relationship to MM2S Stream and Control Stream**

AXI 控制流

从分散聚集描述符向用户设备数据的目标设备提供 AXI 控制流。 控制数据与 MM2S 主数据流相关联，并且可以在主数据包之前，之中或之后从 AXI DMA 中发送出去。 允许目标设备进行节流，并且可能发生 AXI DMA 的节流。 图 2-37 显示了如何在 AXI 控制流上显示描述符用户应用程序字段的示例。

AXI DMA 向目标设备插入一个指示数据类型的标志。 这作为第一个单词发送。 对于以太网，第一个字的四个最高有效位（MSB）中的控制标签为 0xA。

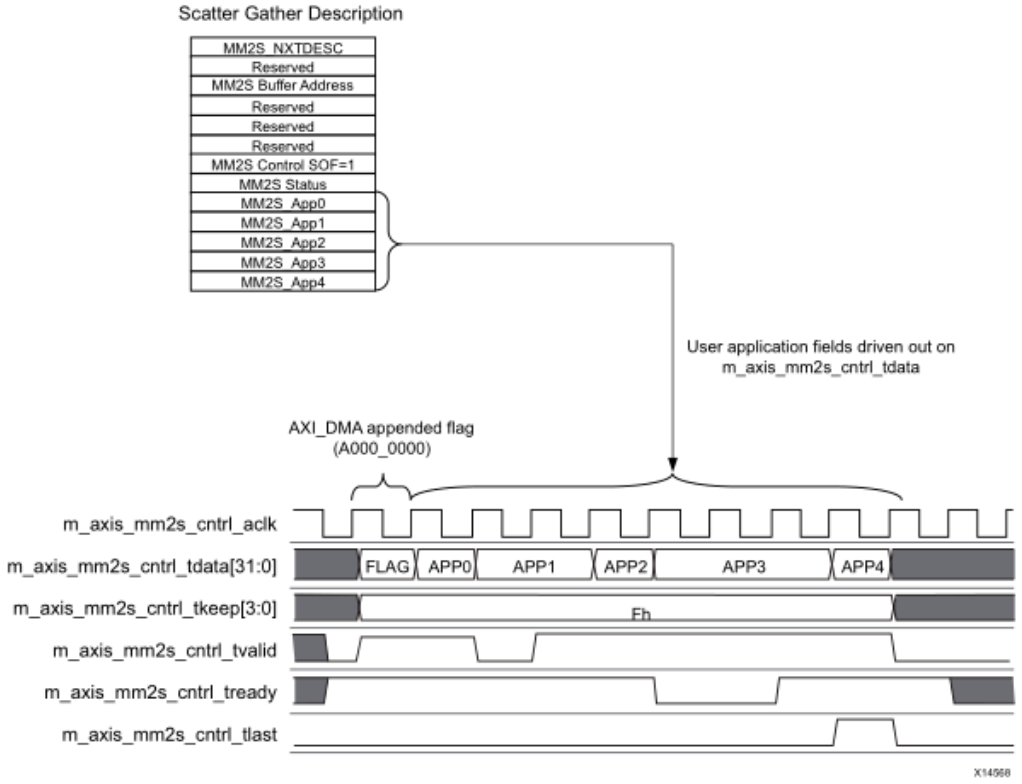

Figure 2-37: **Example User Application Field / Timing for MM2S Control Stream**

S2MM Descriptor Settings and AXI Status Stream

描述符 RXSOF / RXEOF 设置与 AXI 状态流之间的关系如图 2-38 所示。 RXSOF = 1 的描述符描述了包含接收数据包第一部分的缓冲区。 RXEOF = 1 的描述符描述了包含接收数据包最后部分的缓冲区。

为了正确操作，软件必须指定足够的缓冲区空间（描述符链的每个描述符中缓冲区长度的总和），以大于或等于接收到的最大大小的数据包。

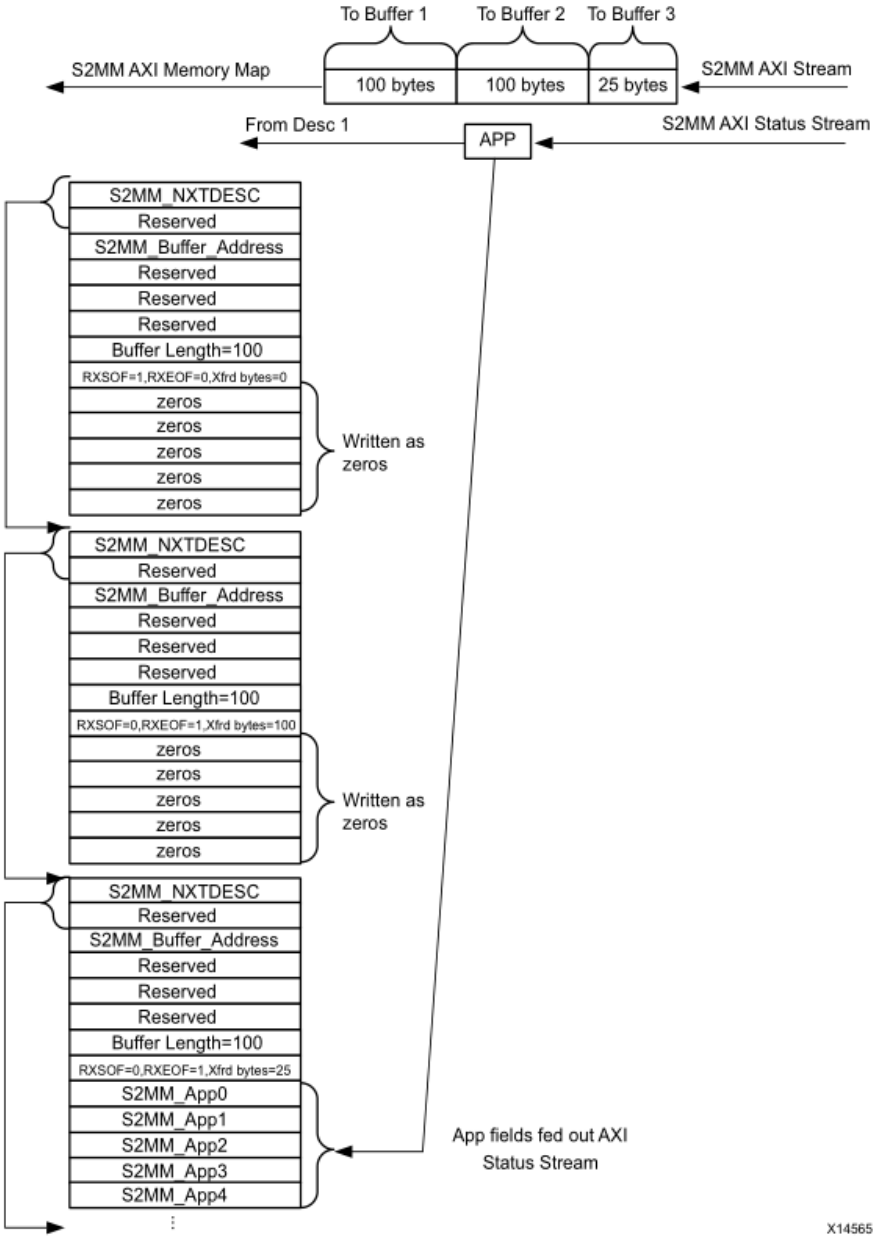

Figure 2-38: **Detail of Descriptor Relationship to S2MM Stream and Status Stream**

如果包括状态/控制流，则将接收到的状态存储在描述符中的用户应用程序字段（APP0 至 APP4）中，并将其设置为 RXEOF。

特定缓冲区的已接收和已存储数据的实际字节数将更新为关联描述符中的“已传输字节”字段。 该软件可以通过将描述符从 RXSOF 移到 RXEOF 并添加“已传输字节数”字段来获得总字节数，从而确定接收到多少字节。 对于在状态流中提供总长度的应用程序，此值存储在描述符中用户定义的应用程序位置中，RXEOF = 1。

AXI 状态流

提供了 AXI 状态流，用于将目标设备状态传输到分散收集描述符中的用户应用程序数据字段。 状态数据与 S2MM 主数据流相关联。 如图 2-39 所示，状态数据包更新为检测到的描述该数据包的最后一个描述符（RXEOF = 1）的 app 字段。 通常，状态流应位于 S2MM 数据流的开头。 如果禁用了“在状态流中使用 RxLength”，则状态流可以在 S2MM 帧的过程中随时出现。 仅当接收到整个状态流时，才会发生帧结束（EOF）缓冲区描述符（BD）更新。

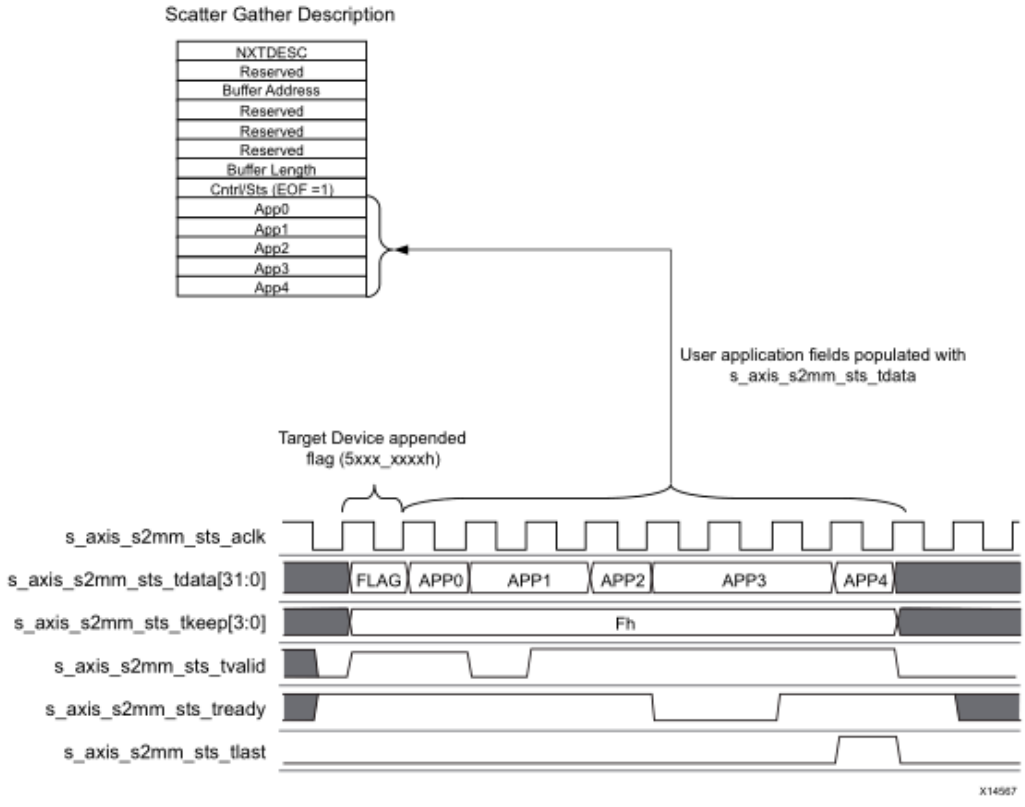

Figure 2-39: **Example User Application Field / Timing for S2MM Status Stream**

多通道 DMA 支持

重要信息：AXI DMA 的多通道支持将停止。 有关多通道的支持，请参见 AXI 多通道直接内存访问（PG288）[参考资料 12]。

多通道模式使 DMA 可以连接到流式传输侧的多个主机和从机。 添加了与源和目标信令关联的一组新信号。 他们是：

•tid – 5 位信号。 用户定义的边带信令。
•tdest – 5 位信号。 提供数据流的粗略路由信息。
•tuser – 4 位信号。 用户定义的边带信令。

散布收集模式（C_INCLUDE_SG = 1）

添加了新的描述符字段以支持多通道和 2D 传输。 如 AXI DMA 多通道操作中所述，AXI DMA 支持有效的二维内存访问模式，可在 AXI4-Stream 通道上传输 2-D 块。 内存访问模式由三个参数控制：HSIZE，VSIZE 和 STRIDE。 通过数据包开始和数据包结束标志支持每个数据包多个描述符。

在这种模式下，Vivado®集成设计环境（IDE）IP 定制功能将禁用状态/控制流。

通过启用多通道模式并选择 MM2S 和 S2MM 路径上所需的通道数，可以将 AXI DMA 设置为多通道模式。

MM2S (TX) Descriptor

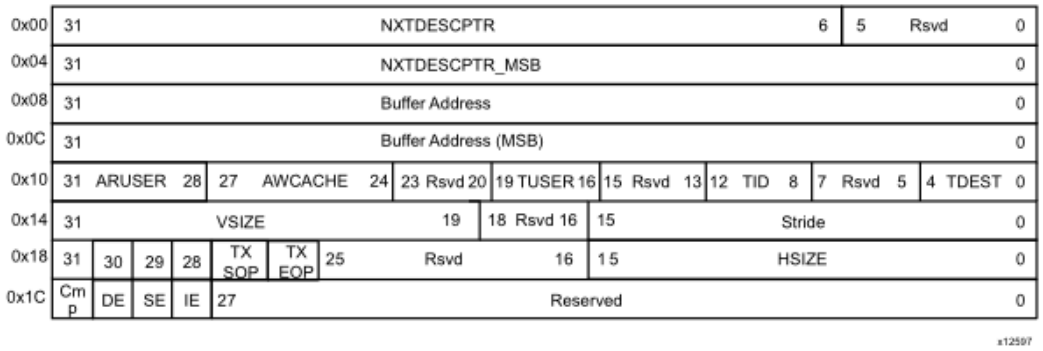

Figure 2-40: **TX Descriptor**

表 2-42：TX 描述符字段

地址偏移	名称	描述
00h	NXTDESC	位 5：0 –保留 位 31：6 –下一个描述符指针
04h	NXTDESC MSB	提供下一个描述符指针的高 32 位。 当 AXI DMA 配置为大于 32 的地址空间时适用。
08h	BUFFER_ADDRESS	位 31：0 –缓冲区地址 提供要从内存映射传输到流的数据的位置。地址应与内存映射数据宽度对齐。
0Ch	BUFFER_ADDRESS	提供缓冲区地址的高 32 位。 这仅在 AXI DMA 配置为大于 32 的地址空间时适用。
		多通道控制位。 位 4：0 – TDEST 提供数据流的路由信息。 TDEST 值对于整个数据包都是静态的。 TX 描述符字段中提供的 TDEST 值呈现在流侧的 TDEST 信号上。 •位 7：5 –保留
10h	MC_CTL	•位 12：8 – TID：提供流标识符。 对于整个数据包，TID 值是静态的。 TX 描述符字段中提供的 TID 值呈现在流传输侧的 TID 信号上。 •位 15:13 –保留 •位 19:16 – TUSER：边带信号，用于用户定义的信息。 对于整个数据包，TUSER 值都是静态的。 TX 描述符字段中提供的 TUSER 值显示在流式端的 TUSER 信号上。 •位 23:20 –保留 •位 27:24 – ARCACHE：缓存类型。 该信号提供有关传输的可缓存特性的其他信息。 有关不同的解码机制，请参见 AMBA®AXI 和 ACE 协议规范[Ref 3]。 来自 TX 描述符的 ARCACHE 值在地址周期内显示在 ARCACHE [3：0]总线上。 该字段的默认值为 0011。 •位 31:28 – ARUSER：用于用户定义信息的边带信号。 来自 TX 描述符的 ARUSER 值显示在 ARUSER [3：0]上。 ARUSER 值及其解释是用户定义的。 通过在链中的所有描述符中编程相同的值，可以使整个包的 ARUSER 保持静态。
14h	STRIDE_VSIZE	•位 15：0 –步幅控制。 它是连续“水平”读取的第一个地址之间的地址距离。 读取将从缓冲区地址开始并读取 HSIZE 字节，然后跳过 STRIDE-HSIZE 地址并读取 HSIZE 字节，依此类推。 这一直持续到读取了 VSIZE 行为止。 在 AXI4-Stream 上，它作为一个连续数据包在 m_axis_mm2s_接口上发送出去，并在传输的最后一个数据节拍上以 TLAST 的单个声明终止。 •位 18:16 –保留 •位 31:19 –跨步访问的“水平线”数。 可以表示二维视频数据或二维矩阵的大小。 这是预期每个数据包传输的传输次数（每个 HSIZE 字节长）。
18h	HSIZE	•位 15：0 –从连续的连续字节地址开始在每个“水平行”中传输的字节数。 当以行主要顺序读取矩阵时，可以表示视频行的一部分或矩阵行的一部分。 •位 25:16 –保留 •位 26 – TXEOP –数据包结束标志。 它指示与此描述符关联的缓冲区最后发送。 该标志由 CPU 设置。 0 –不是数据包结尾。 1 –数据包结束 •位 27 – TXSOP –数据包开始标志。 它指示与此描述符关联的缓冲区首先被发送。 该标志由 CPU 设置。MC_STS 0 –不开始数据包。 1 –数据包开 •位 31:28 –保留
1Ch	MC_STS	多通道状态位。 •位 27：0 –保留•位 28-IE-由于欠载或超载情况导致的 DMA 内部错误。 0 –没有 DMA 内部错误 1 –检测到 DMA 内部错误。 DMA 引擎停止。 •位 29 – SE – DMA 从站错误。 如果从内存映射接口读取的从设备发出从设备错误，则会发生此错误。 0 –无 DMA 从站错误 1 –检测到 DMA 从站错误。 DMA 引擎停止 •位 30 – DE – DMA 解码错误。 如果地址请求的地址无效，则会发生此错误。 0 –没有 DMA 解码错误 1 –检测到 DMA 解码错误。 DMA 引擎停止 •位 31 – Cmp –已完成。 这向软件指示 DMA 引擎已完成传输。 0 –描述符未完成 1 –描述符已完成

注意：

1.从 AXI 读取角度来看，ARCACHE，ARUSER 值很重要。 这些值应根据需要在描述符中指定。 对于正常操作，应将 ARCACHE 设置为 0011，而将 ARUSER 设置为 0000。

2. VSIZE 上的 0 值是非法的，并导致多通道 DMA 无法正常工作。

S2MM（RX）描述符

S2MM (RX) Descriptor

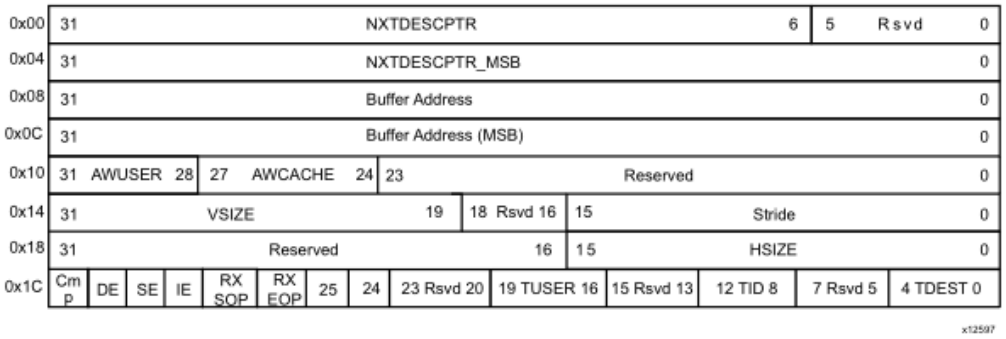

Figure 2-41: **RX Descriptor**

表 2-43：RX 描述符字段

地址偏移	名称	描述
00h	NXTDESC	位 5：0 –保留 位 31：6 –下一个描述符指针
04h	NXTDESC_MSB	提供下一个描述符指针的高 32 位。当 DMA 配置为大于 32 的地址空间时适用。
08h	BUFFER_ADDRESS	位 31：0 –缓冲区地址 提供可用于存储从流传输到内存映射的数据的缓冲区空间的位置。该地址应与内存映射数据宽度对齐。
0Ch	BUFFER_ADDRESS_ MSB	提供缓冲区地址的高 32 位。 仅当 AXI DMA 配置为大于 32 的地址空间时使用。
10h	CACHE_USER_CTL	位 23：0 –保留 位 27:24 – AWCACHE –缓存类型。 该信号提供有关传输的可缓存特性的其他信息。 有关不同的解码机制，请参见 AMBA AXI 和 ACE 协议规范[Ref 3]。 在地址周期期间，来自 RX 描述符的 AWCACHE 值显示在 AWCACHE [3：0]总线上。 此字段的默认值应为 0011。 位 31:28 – AWUSER –用于用户定义信息的边带信号。 来自 RX 描述符的 AWUSER 值显示在 AWUSER [3：0]上。 AWUSER 值及其解释是用户定义的。 通过在链中的所有描述符中编程相同的值，可以使整个包的 AWUSER 保持静态。
14h	STRIDE_VSIZE	位 15：0 –步幅控制。 它是连续“水平”写入的第一个地址之间的地址距离。 写操作从缓冲区地址开始，并写入 HSIZE 字节，然后跳过 STRIDE-HSIZE 地址，并写入 HSIZE 字节，依此类推。 这一直持续到写入 VSIZE。 在 AXI4-Stream 上，它作为一个连续数据包在 s_axis_s2mm_接口上接收，并在传输的最后一个数据节拍上以 TLAST 的单个声明终止。 位 18:16 –保留 位 31:19 –用于跨步访问的“水平线”数。 可以表示二维视频数据或二维矩阵的大小。 VSIZE 传输数，每个 HSIZE 字节长，预计将为每个数据包接收。
18h	HSIZE	位 15：0 –从连续的连续字节地址开始的每个“水平行”中要传输的字节数。 当矩阵以行主要顺序存储时，可以表示视频行的一部分或矩阵行的一部分。 位 31:16 –保留
1Ch	MC_STS	多通道状态位。 位 4：0 – TDEST 提供数据流的路由信息。 TDEST 值对于整个数据包都是静态的。 TDEST 值是从传入流中捕获的，并在此字段中进行更新。 位 7：5 –保留 位 12：8 – TID 提供流标识符。 对于整个数据包，TID 值是静态的。 TID 值是从传入流中捕获的，并在此字段中进行更新。 位 15:13 –保留 位 19:16 – TUSER –用于用户定义信息的边带信号。 对于整个数据包，TUSER 值都是静态的。 从输入流中捕获 TUSER 值，并在此字段中更新。 位 25:20 –保留 位 25:24 –保留

		位 26 – RXEOP –数据包结束标志。 它指示与此描述符关联的缓冲区包含数据包的最后部分。 该标志由 AXI DMA 设置。 •0 –不结束包 •1 –结束包 位 27 – RXSOP –数据包开始标志。 它指示与此描述符关联的缓冲区包含数据包的开始。 该标志由 AXI DMA 设置。 •0 –不开始数据包 •1 –开始数据包 位 28 – IE –由于欠载或超载情况导致的 DMA 内部错误。 •0-没有 DMA 内部错误 •1-检测到 DMA 内部错误。 DMA 引擎停止。 位 29 – SE – DMA 从站错误。 如果从内存映射接口读取的从设备发出从设备错误，则会发生此错误。 •0-无 DMA 从站错•1-检测到 DMA 从站错误。 DMA 引擎停止。 位 30 – DE – DMA 解码错误。 如果地址请求的地址无效，则会发生此错误。 •0-没有 DMA 解码错误 •1-检测到 DMA 解码错误。 DMA 引擎停止。 位 31 – Cmp –已完成。 这向软件指示 DMA 引擎已完成传输。 •0 –描述符未完成 •1 –描述符已完成

注意：

1. AWCACHE，AWUSER 值对 AXI 写预期很重要。 这些值应根据需要在描述符中指定。对于正常操作，应将 AWCACHE 设置为 0011，而将 AWUSER 设置为 0000。

2. VSIZE 上的值“ 0"是非法的，并导致多通道 DMA 无法正常工作。

AXI DMA 多通道操作

本节描述了 MM2S 端和 S2MM 端的描述符以及相关数据的端到端控制和数据流。

MM2S

MM2S 与正常的 AXI DMA 操作相似。通过软件对 MM2S_CURDESC 和 MM2S_TAILDESC 进行编程时，AXI DMA 将获取描述符链并进行处理，直到到达尾部描述符。在 AXI DMA 中，假设描述符中定义的整个数据包的 TDEST，TID 和 TUSER 字段保持不变。 也就是说，在由 TDEST，TID，TUSER 定义的逻辑通道上进行的每个数据包传输都将在 DMA 传输另一个数据包之前完成。 尽管可以交错处理多个通道的数据包传输，但是在启动之后，每个通道都必须运行到完成才能进行另一次传输。 在这种假设下，避免死锁是您的责任。 如果描述符内的（TDEST，TID，TUSER）字段不遵守这些假设，则 AXI DMA 不会发出错误状态信号。 保持软件的一致性取决于软件。

TX 描述符包含多通道模式下的控制和状态字段。 对于事务结束时传输的字节数，此描述符上没有状态更新。

错误信息与失败描述符的当前描述符指针一起被捕获到寄存器中。 可以通过查询 MM2S_DMASR 的 IDLE 位或通过在 MM2S_DMACR 中启用它的中断来知道链传输的完成情况。

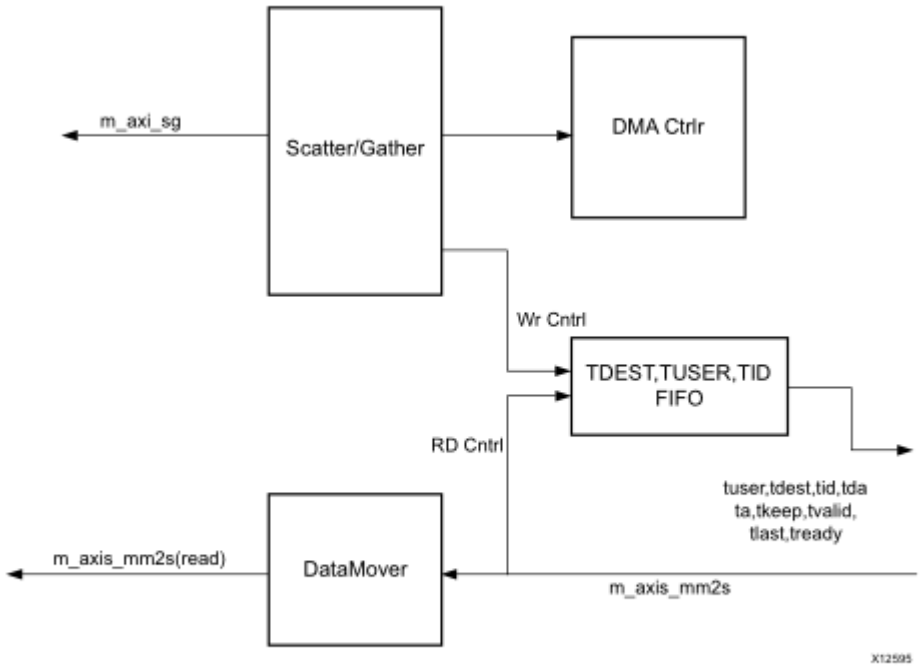

Figure 2-42: **MM2S Control and Datapath Flow**

S2MM

从输入流中采样 tdest 信号。 DMA 控制器使用此值来读取相应的 S2MM_CURDESC 和 S2MM_TAILDESC 并将其提供给 Scatter Gather 模块。 反过来，该 SG 模块获取该 tdest 的描述符链。 这意味着应在数据包到达 S2MM 通道之前完成（对于所有通道）缓冲区描述符链的设置。 在数据包/数据到达之前，应对当前描述符和尾部描述符寄存器进行编程。 AXI DMA 通过取消踩踏来保持流数据，直到获取相应的描述符为止。 然后，AXI DMA 写操作从缓冲区地址开始，一直持续到从流传输端接收到 tlast。 对于特定通道，当数据传输完成时，当前描述符寄存器将加载该链中下一个描述符的地址。 这是为了确保正确服务下一个数据包。 您必须确保正确设置了描述符链，并且正确处理了尾部描述符，以避免当前描述符的跳转。

TDEST，TID，TUSER 值从传入流中捕获并内部存储。 这些值不应在数据包中间更改。在该描述符的数据写入完成以及与其他状态位一起写入存储器映射侧后，将在链的每个描述符中更新这些值。

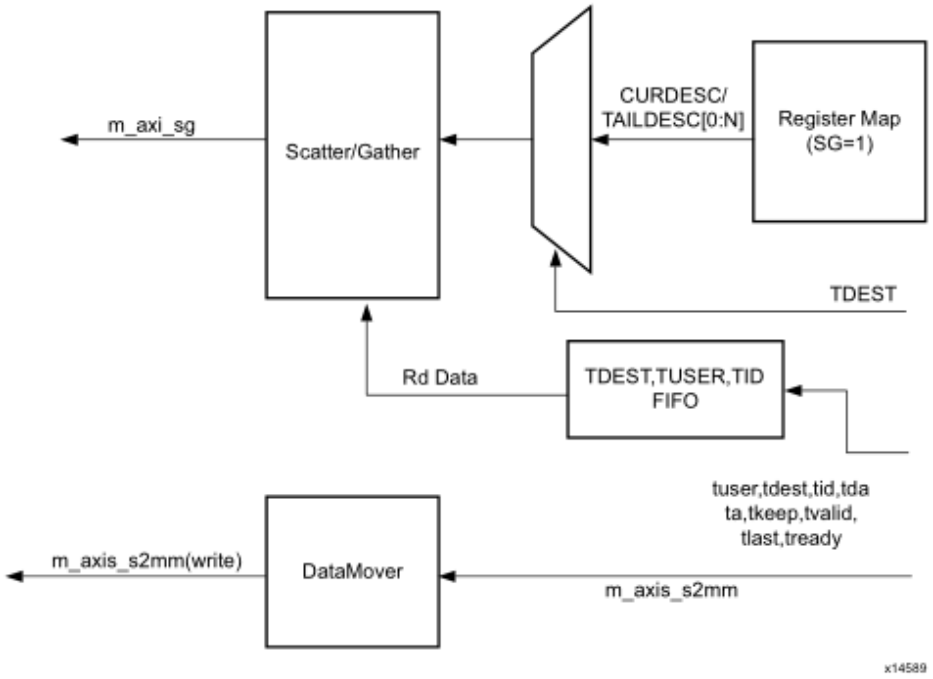

Figure 2-43: **S2MM Control and Datapath Flow**

二维转移

在多通道模式下，AXI DMA 支持 2-D 存储器访问模式，可通过 AXI4-Stream 通道进行有效传输。

访问模式由描述符字段 HSIZE，VSIZE 和 STRIDE 控制，这些字段允许在（隐式）2-D 数组中传输子块。 在连续的“行”子块的起始地址之间指定 HSIZE。 对于二维传输，HSIZE，VISZE 和 STRIDE 应该按字节对齐。 HSIZE，VSIZE 或 STRIDE 的值未对齐会导致意外的行为。

每个读（MM2S）或写（S2MM）传输均由 VSIZE 传输组成，每个传输的大小均为 HSIZE。

每次连续传输的起始地址都是从上一次传输的起始地址开始的 STRIDE 地址（最初是数据包传输的 BaseAddr）。

图 2-44 显示了二维数据格式的示例。

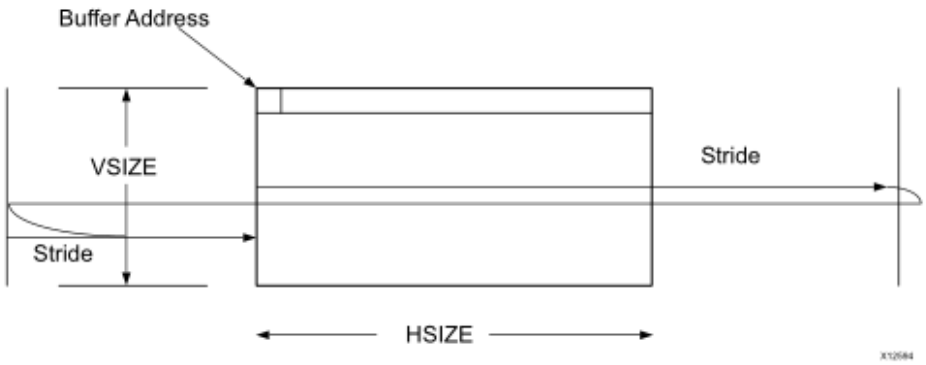

Figure 2-44: **2-D Data Format**

MM2S

读取从缓冲区地址开始，由 VSIZE 读取突发组成，每个突发都有 HSIZE 字节。 每个读取突发的起始地址的 STRIDE 地址大于先前读取突发的起始地址。

示例：缓冲区地址= 08，VSIZE = 06，HSIZE = 256 字节，步幅= 512 字节在这种情况下，读取从缓冲区地址位置 08 开始并继续读取 HSIZE（256）字节。 第二行从缓冲区地址 + Stride = 512 + 8 = 520 开始。它将继续读取 HSIZE（256）字节。 第三行开始于 520 + 512 = 1,032，第四行开始于 1032 + 512 = 1,544。 VSIZE 行以这种模式继续读取。

在 AXI4-Stream 上，它作为一个连续数据包在 m_axis_mm2s 接口上传输，并在传输的最后一个数据节拍上以 tlast 的断言终止。

S2MM

写操作从缓冲区地址开始，由 VSIZE 写突发组成，每个突发都有 HSIZE 字节。 每个写突发的起始地址的 STRIDE 地址大于上一个突发写的起始地址。 在 AXI4-Stream 接口上，此消息作为一个连续数据包在 s_axis_s2mm_接口上接收，并在传输的最后一个数据节拍上通过一次 tlast 断言终止。 到达包的大小应与“缓冲区描述符”中编程的大小匹配。

多通道模式的局限性

•不支持多通道模式的 S2MM 路径中的描述符排队
•不支持 S2MM 路径的小数据包大小（4 个或更少数据拍的背对背数据包）
•对于 2D 访问，地址和 Hsize 必须与 Memory Map 数据宽度对齐。

多通道寄存器映射（SG = 1）

寄存器映射被修改为在 S2MM 路径上最多支持 16 个 TDEST。

00	MM2S_DMACR
04	MM2S_DMASR
08	MM2S_CURDESC0
0C	MM2S_CURDESC0_MSB
10	MM2S_TAILDESC0
14	MM2S_TAILDESC0_MSB
18d	Reserved
1C	Reserved
20	Reserved
24	Reserved
28	Reserved
2C	SG_CTL
30	S2MM_DMACR
34	S2MM_DMASR
38	S2MM_CURDESC0
3C	S2MM_CURDESC0_MSB
40	S2MM_TAILDESC0
44	S2MM_TAILDESC0_MSB
48	Reserved
4C	Reserved
50	Reserved
54	Reserved
58	Reserved
5C	Reserved
60	Reserved
64	Reserved
68	Reserved
6C	Reserved
70	S2MM_CURDESC1
74	S2MM_CURDESC1_MSB
78	S2MM_TAILDESC1
7C	S2MM_TAILDESC1_MSB
80	Reserved
84	Reserved
88	Reserved
8C	Reserved
90	S2MM_CURDESC2
94	S2MM_CURDESC2_MSB
98	S2MM_TAILDESC2
9C	S2MM_TAILDESC2_MSB
A0	Reserved
A4	Reserved
AB	Reserved
AC	Reserved
B0	S2MM_CURDESC3
B4	S2MM_CURDESC3_MSB
B8	S2MM_TAILDESC3
BC	S2MM_TAILDESC3_MSB

C0	Reserved
C4	Reserved
C8	Reserved
CC	Reserved
D0	S2MM_CURDESC4
D4	S2MM_CURDESC4_MSB
D8	S2MM-TAILDESC4
DC	S2MM_TAILDESC4_MSB
E0	Reserved
E4	Reserved
E8	Reserved
EC	Reserved
F0	S2MM_CURDESC5
F4	S2MM_CURDESC5_MSB
F8	S2MM_TAILDESC5
FC	S2MM_TAILDESC5_MSB
100	Reserved
104	Reserved
108	Reserved
10C	Reserved
110	S2MM_CURDESC6
114	S2MM_CURDESC6_MSB
118	S2MM_TAILDESC6
11C	S2MM_TAILDESC6_MSB
120	Reserved
124	Reserved
128	Reserved
12C	Reserved
130	S2MM_CURDESC7
134	S2MM_CURDESC7_MSB
138	S2MM_TAILDESC7
13C	S2MM_TAILDESC7_MSB
140	Reserved
144	Reserved
148	Reserved
14C	Reserved
150	S2MM_CURDESC8
154	S2MM_CURDESC8_MSB
158	S2MM_TAILDESC8
15C	S2MM_TAILDESC8_MSB
160	Reserved
164	Reserved
168	Reserved
16C	Reserved
170	S2MM_CURDESC9
174	S2MM_CURDESC9_MSB
178	S2MM_TAILDESC9
17C	S2MM_TAILDESC9_MSB

180	Reserved
184	Reserved
188	Reserved
18C	Reserved
190	S2MM_CURDESC10
194	S2MM_CURDESC10_MSB
198	S2MM-TAILDESC10
19C	S2MM_TAILDESC10_MSB
1A0	Reserved
1A4	Reserved
1A8	Reserved
1AC	Reserved
1B0	S2MM_CURDESC11
1B4	S2MM_CURDESC11_MSB
1B8	S2MM_TAILDESC11
1BC	S2MM_TAILDESC11_MSB
1C0	Reserved
1C4	Reserved
1C8	Reserved
1CC	Reserved
1D0	S2MM_CURDESC12
1D4	S2MM_CURDESC12_MSB
1D8	S2MM_TAILDESC12
1DC	S2MM_TAILDESC12_MSB
1E0	Reserved
1E4	Reserved
1E8	Reserved
1EC	Reserved
1F0	S2MM_CURDESC13
1F4	S2MM_CURDESC13_MSB
1F8	S2MM_TAILDESC13
1FC	S2MM_TAILDESC13_MSB
200	Reserved
204	Reserved
208	Reserved
20C	Reserved
210	S2MM_CURDESC14
214	S2MM_CURDESC14_MSB
218	S2MM_TAILDESC14
21C	S2MM_TAILDESC14_MSB
220	Reserved
224	Reserved
228	Reserved
22C	Reserved
230	S2MM_CURDESC15
234	S2MM_CURDESC15_MSB
238	S2MM_TAILDESC15
23C	S2MM_TAILDESC15_MSB

X12591

Figure 2-45: **Register Map for Multichannel Support in SG = 1**

核心设计

典型系统互连

AXI DMA 内核设计为通过用户系统中的 AXI 互连进行连接。 典型的 MicroBlaze™处理器配置如图 3-1 所示。系统微处理器可以通过 AXI4-Lite 接口访问 AXI DMA。集成的 Scatter / Gather Engine 从系统内存中获取缓冲区描述符，然后协调 AXI IP 和 DDRx 之间的主要数据传输。 可选的控制和状态流向与基于以太网的 IP 之间提供与分组相关的信息，例如校验和卸载控制/状态。 AXI DMA 内核的双中断输出被路由到系统中断控制器。

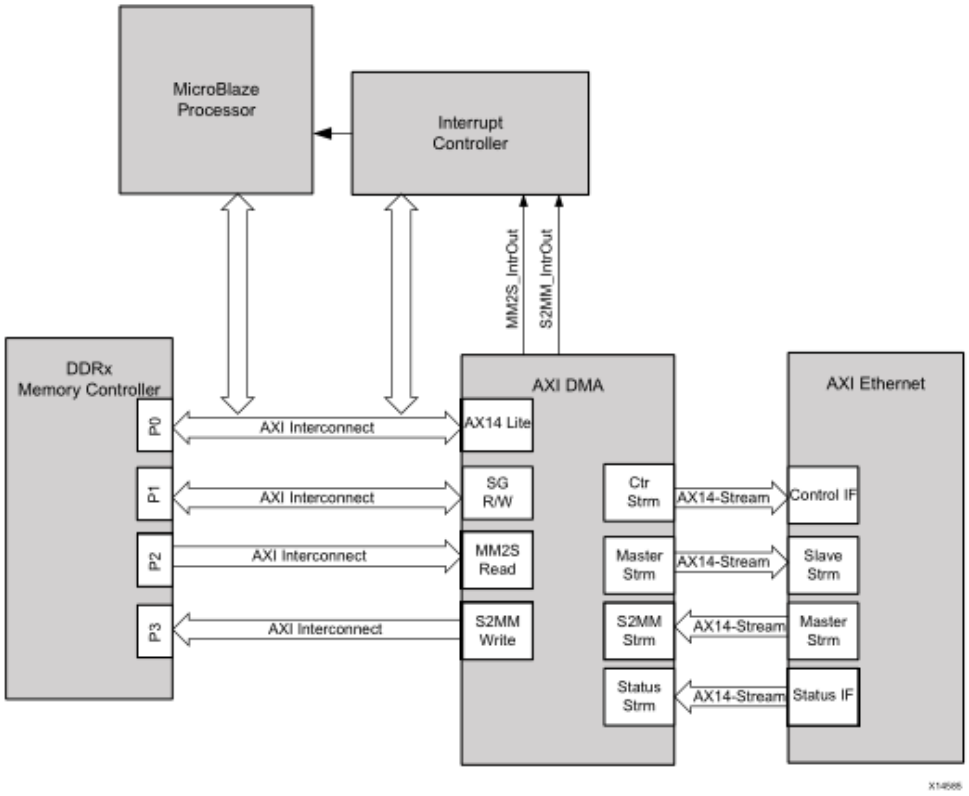

Figure 3-1: **Typical MicroBlaze Processor System Configuration (AXI Ethernet)**

除基于以太网的 AXI IP 外，AXI DMA 内核还可以连接到用户系统。 控制和状态流是可选的，并且只能与基于以太网的 IP 内核一起使用。

注意：

在没有任何设置的情况下（也就是说，在对其进行编程之前），AXI DMA 将在获取四拍流数据后将 s_axis_s2mm_tready 信号拉低。 这将限制输入数据流。 为了尽可能减少节流，请确保将 AXI DMA 设置为在实际数据到达之前就可以运行很多。

时钟

有四个时钟输入：

- 用于 MM2S 接口的 m_axi_mm2s_aclk
- 用于 S2MM 接口的 m_axi_s2mm_aclk
- 用于 AXI4-Lite 控制接口的 s_axi_lite_aclk
- 用于分散收集接口的 m_axi_sg_clk

AXI DMA 提供两种时钟操作模式：异步和同步。

设置启用异步时钟将启用异步模式并创建四个时钟域。 这使高性能用户可以以比 DMA 控制（例如 AXI4-Lite 接口，SG 引擎，DMA 控制器）更高的时钟速率运行主数据路径，从而有助于 FPGA 的放置和时序。

在同步模式下，所有逻辑都在单个时钟域中运行。 m_axi_sg_aclk，m_axi_mm2s_aclk 和 m_axi_s2mm_aclk 必须绑定到相同的源，s_axi_lite_aclk 可以连接到较慢的时钟。 在异步模式下，时钟可以异步运行，但是 s_axi_lite_aclk 必须小于或等于 m_axi_sg_aclk，并且 m_axi_sg_aclk 必须小于或等于 m_axi_mm2s_aclk 或 m_axi_s2mm_aclk 中的较慢者。

表 3-1 列出了异步模式下信号集及其对应时钟之间的关系。

表 3-1：异步模式时钟分配

Table 3-1: **Asynchronous Mode Clock Distribution**

Clock Source	I/O Ports (Scatter Gather Enabled)	I/O Ports (Scatter Gather Disabled)
s_axi_lite_aclk	All s_axi_lite_* Signals mm2s_introut s2mm_introut axi_resetn	All s_axi_lite_* Signals mm2s_introut s2mm_introut axi_resetn
m_axi_sg_aclk	All m_axi_sg_* Signals	N/A
m_axi_mm2s_aclk	All m_axi_mm2s_* Signals All m_axis_mm2s_* Signals mm2s_prmry_reset_out_n mm2s_cntrl_reset_out_n	All m_axi_mm2s_* Signals All m_axis_mm2s_* Signals mm2s_prmry_reset_out_n
m_axi_s2mm_aclk	All m_axi_s2mm_* Signals All s_axis_s2mm_* Signals s2mm_prmry_reset_out_n s2mm_sts_reset_out_n	All m_axi_s2mm_* Signals All s_axis_s2mm_* Signals s2mm_prmry_reset_out_n

复位

axi_resetn 信号需要断言至少 16 个最慢的时钟周期，并且需要与 s_axi_lite_aclk 同步。

编程序列

直接寄存器模式（简单 DMA）

直接寄存器模式（禁用了 Scatter Gather Engine）提供了一种配置，用于在 MM2S 和 S2MM 通道上执行简单的 DMA 传输，而这需要较少的 FPGA 资源利用率。通过访问 DMACR，源或目标地址和长度寄存器来启动传输。 传输完成后，DMASR.IOC_Irq 会为关联的通道断言，如果使能，则会产生中断输出。

MM2S 通道的 DMA 操作是按以下顺序设置和启动的：

1.通过将运行/停止位设置为 1（MM2S_DMACR.RS = 1），开始运行 MM2S 通道。 停止位（DMASR.Halted）应置为无效，指示 MM2S 通道正在运行。

2.如果需要，通过向 MM2S_DMACR.IOC_IrqEn 和 MM2S_DMACR.Err_IrqEn 写 1 来使能中断。当 AXI DMA 配置为直接寄存器模式时，不使用延迟中断，延迟计数和阈值计数。

3.将有效的源地址写入 MM2S_SA 寄存器。 如果将 AXI DMA 配置为大于 32 的地址空间，则对 MM2S_SA MSB 寄存器进行编程。 如果未为数据重新对齐配置 AXI DMA，则必须对齐有效地址，否则将产生未定义的结果。 被认为是对齐还是未对齐的是基于流数据宽度的。在 Micro Mode 下配置 AXI_DMA 时，您有责任指定正确的地址。 Micro DMA 不会处理 4K 边界。

例如，如果“内存映射数据宽度” = 32，则数据位于字偏移量（32 位偏移量）处即 0x0、0x4、0x8、0xC 等，则对齐。 如果启用了 DRE 并且流数据宽度<128，则源地址可以具有任何字节偏移量。

4.在 MM2S_LENGTH 寄存器中写入要传输的字节数。 写入的零值无效。 非零值会导致 MM2S_LENGTH 字节数在 MM2S AXI4 接口上读取并从 MM2S AXI4-Stream 接口发送出去。MM2S_LENGTH 寄存器必须最后写入。 所有其他 MM2S 寄存器可以任意顺序写入。 对于 Micro DMA，此值不能超过[Burst_length *（内存映射的数据宽度）/ 8]。

按照以下顺序设置和启动 S2MM 通道的 DMA 操作：

1. 通过将运行/停止位设置为 1（S2MM_DMACR.RS = 1），启动 S2MM 通道运行。 停止位（DMASR.Halted）应置为无效，指示 S2MM 通道正在运行。

2.如果需要，通过将 1 写入 S2MM_DMACR.IOC_IrqEn 和 S2MM_DMACR.Err_IrqEn 来使能中断。 当 AXI DMA 配置为直接寄存器模式时，不使用延迟中断，延迟计数和阈值计数。

3.将有效的目标地址写入 S2MM_DA 寄存器。如果将 AXI DMA 配置为大于 32 的地址空间，则对 S2MM_DA MSB 寄存器进行编程

4.如果未为数据重新对齐配置 AXI DMA，则必须对齐有效地址，否则将产生未定义的结果。被认为是对齐还是未对齐的是基于流数据宽度的。

例如，如果“内存映射数据宽度”为 32，则如果数据位于字偏移量（32 位偏移量）（即 0x0、0x4、0x8、0xC 等）处，则对齐数据。 如果启用了 DRE 并且流数据宽度<128，则目标地址可以具有任何字节偏移量。

5.将接收缓冲区的长度（以字节为单位）写入 S2MM_LENGTH 寄存器。 零值无效。 非零值会导致在 S2MM AXI4 接口上写入在 S2MM AXI4-Stream 接口上接收的字节数。 大于或等于最大接收数据包的值必须写入 S2MM_LENGTH。 小于接收到的字节数的接收缓冲区长度值会产生不确定的结果。
在 Micro 模式下配置 AXI DMA 时，该值应与 S2MM AXI4-Stream 接口上接收的字节完全匹配。 S2MM_LENGTH 寄存器必须最后写入。 所有其他 S2MM 寄存器可以任意顺序写入。

分散/聚集模式

AXI DMA 操作需要一个驻留内存的数据结构，该结构保存要执行的 DMA 操作列表。 该指令列表被组织成所谓的描述符链。每个描述符都有一个指向要处理的下一个描述符的指针。然后，链中的最后一个描述符指向链中的第一个描述符。

分散收集操作允许一个包由多个描述符描述。 此功能的典型用法是允许从内存中的某个位置存储或获取标头，并从另一位置存储或获取有效负载数据。 利用此功能的软件应用程序可以提高吞吐量。 为了描述缓冲区描述符链中的数据包，使用了帧起始位（TXSOF）和帧结束位（TXEOF）。 当 DMA 通过 TXSOF 位置 1 获取描述符时，将触发数据包的开始。 数据包继续获取后续描述符，直到获取 TXEOF 位置 1 的描述符为止。

在开始接收数据包时，在接收（S2MM）通道上，AXI DMA 用 RXSOF 标记描述符，向软件指示与该描述符关联的数据缓冲区包含数据包的开头。 如果接收到的数据包的字节数比描述符中指定的字节数长，则使用下一个描述符缓冲区来存储接收数据包的其余部分。 继续进行此提取和存储过程，直到整个接收包都已传输。 当接收到数据包末尾时，正在处理的描述符由 AXI DMA 标记为 RXEOF = 1。 这向软件指示与该描述符关联的缓冲区包含数据包的末尾。

每个描述符的状态字段包含为该特定描述符实际传输的字节数。 该软件可以通过从 RXSOF 描述符经过描述符链到 RXEOF 描述符来确定为接收数据包传输的字节总数。散点图收集器继续获取一个额外的描述符并存储。

此过程在很大程度上提高了 DMA 性能。

分散收集操作始于设置控制寄存器和描述符指针。

MM2S 通道的 DMA 操作是按照以下顺序设置和启动的：

1.将起始描述符的地址写入当前描述符寄存器。 如果将 AXI DMA 配置为大于 32 的地址空间，则还要对当前描述符的 MSB 32 位进行编程。

2.通过将运行/停止位设置为 1（MM2S_DMACR.RS = 1），开始运行 MM2S 通道。

停止位（DMASR.Halted）应置为无效，指示 MM2S 通道正在运行。

3.如果需要，通过向 MM2S_DMACR.IOC_IrqEn 和 MM2S_DMACR.Err_IrqEn 写 1 来使能中断。

4.将有效地址写入尾部描述符寄存器。 如果将 AXI DMA 配置为大于 32 的地址空间，则还要对尾部描述符的 MSB 32 位进行编程。

5.写入尾描述符寄存器将触发 DMA 开始从内存中获取描述符。 在多通道配置的情况下，当数据包到达 S2MM 通道时开始获取描述符。

6.处理获取的描述符，从内存中读取数据，然后将其输出到 MM2S 流通道。

通过以下顺序设置和启动针对 S2MM 通道的 DMA 操作：
1.将起始描述符的地址写入当前描述符寄存器。 如果将 AXI DMA 配置为大于 32 的地址空间，则还要对当前描述符的 MSB 32 位进行编程。

2.通过将运行/停止位设置为 1（S2MM_DMACR.RS = 1），开始运行 S2MM 通道。

停止位（DMASR.Halted）应置为无效，指示 S2MM 通道正在运行。

3.如果需要，通过将 1 写入 S2MM_DMACR.IOC_IrqEn 和 S2MM_DMACR.Err_IrqEn 来使能中断。

4.将有效地址写入尾部描述符寄存器。 如果将 AXI DMA 配置为大于 32 的地址空间，则还要对当前描述符的 MSB 32 位进行编程。

5.写入尾描述符寄存器将触发 DMA 开始从内存中获取描述符。

6.处理获取的描述符，并将从 S2MM 流通道接收的任何数据写入内存。

循环 DMA 模式

通过对缓冲区描述符（BD）链设置进行某些更改，可以以循环模式运行 AXI DMA。 在循环模式下，DMA 无需中断即可获取和处理相同的 BD。

DMA 继续获取和处理，直到停止或重置为止。 要启用循环操作，应按图 3-2 所示设置 BD 链。

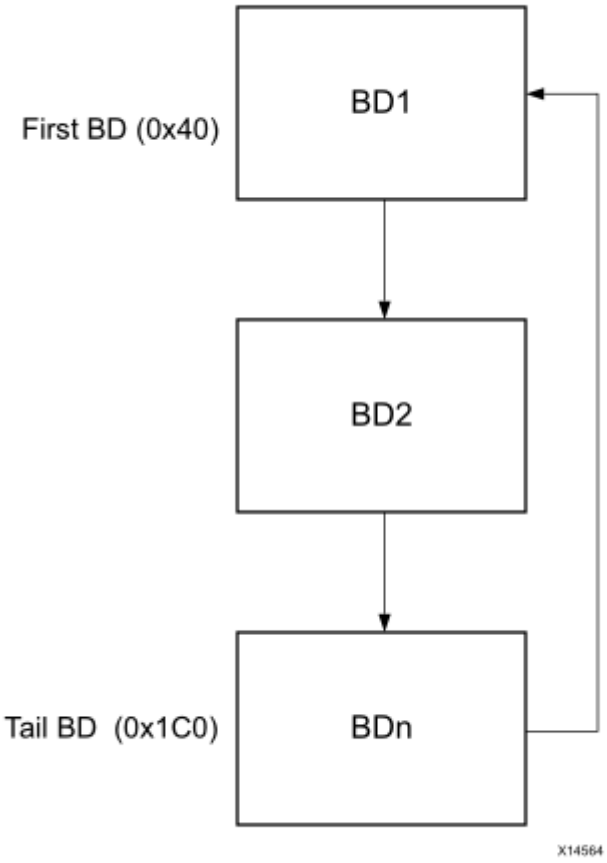

Figure 3-2: **BD Chain**

在此设置中，尾部 BD 指向第一个 BD。 尾部描述符寄存器没有任何作用，仅用于触发 DMA 引擎。 遵循分散/聚集模式中提到的相同编程顺序。 确保控制寄存器中的循环位已设置。

用不属于 BD 链的一部分的值对 Tail Descriptor 寄存器进行编程。 例如说 0x50。

设置尾部描述符寄存器后，DMA 开始获取和处理 BD（以环形方式设置），直到 DMA 停止或复位为止。

设计流程步骤

本章介绍了定制和生成内核，约束内核以及该 IP 内核特有的仿真，综合和实现步骤。以下标准 Vivado Design Suite 用户指南中提供了有关标准 Vivado®设计流程和 IP 集成器的更多详细信息：

- Vivado Design Suite User Guide: Designing with IP (UG896) [Ref 1]
- Vivado Design Suite User Guide: Getting Started (UG910) [Ref 4]
- Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator (UG994) [Ref 5]
- Vivado Design Suite User Guide: Logic Simulation (UG900) [Ref 6]

定制和生成核心

您可以在 Vivado IP 目录中的以下位置找到 AXI DMA。

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

要访问 AXI DMA，请执行以下操作：

1.使用 Vivado 设计工具打开现有项目或创建新项目。

2.打开 IP 目录并导航到任何分类法。

3.双击 AXI Direct Memory Access，打开 AXI DMA Customize IP 窗口。

有关详细信息，请参见《 Vivado 设计套件用户指南：使用 IP 设计（UG896）[参考 1]和《 Vivado Design Suite 用户指南：入门》（UG910）[参考 4]。

注意：本章中的图是 Vivado 集成设计环境（IDE）的图示。

此布局可能与当前版本有所不同。

如果要在 Vivado IP 集成器中自定义和生成内核，请参阅《 Vivado 设计套件用户指南：使用 IP 集成器设计 IP 子系统（UG994）[参考 5]》以获取详细信息。 如本节所述，在验证或生成设计时，Vivado IDE 可能会自动计算某些配置值。 成功完成 validate_bd_design 命令后，您可以查看参数值。

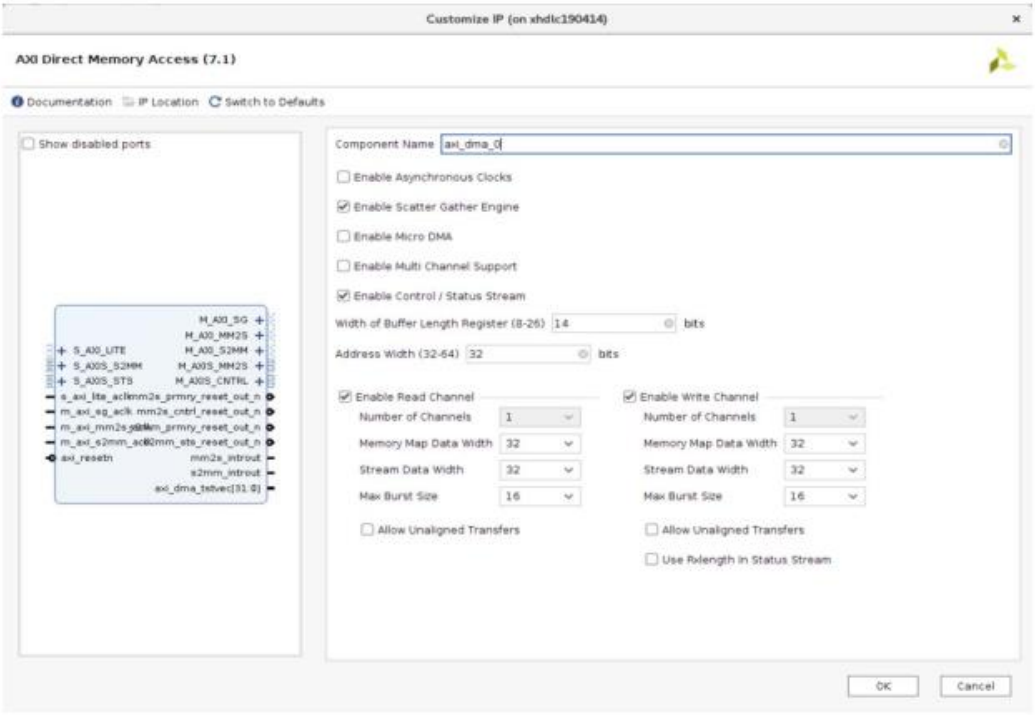

Figure 4-1: **Customize IP Options**

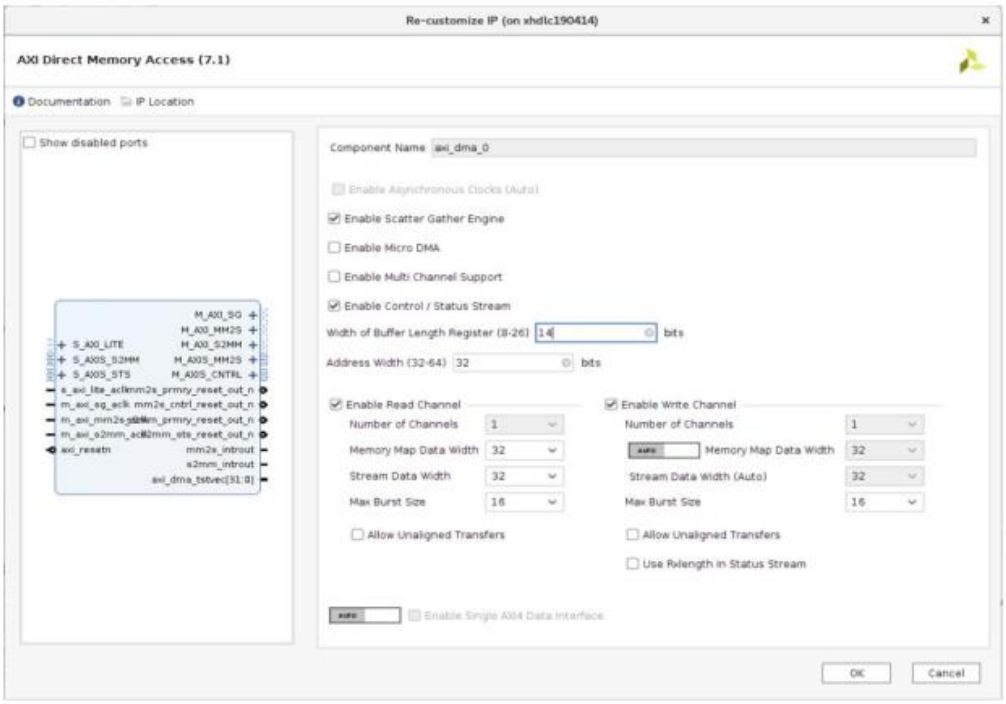

Figure 4-2: **IP Integrator**

栏位说明

组件名称

为核心生成的输出文件的基本名称。 名称必须以字母开头，并且可以由以下任意字符组成：a 到 z，0 到 9 和“ _”。

启用异步时钟

此设置提供了彼此异步操作 MM2S 接口 m_axi_mm2s_aclk，S2MM 接口 m_axi_s2mm_aclk，AXI4-Lite 控制接口 s_axi_lite_aclk 和分散收集接口 m_axi_sg_aclk 的能力。启用异步时钟后，s_axi_lite_aclk 的频率必须小于或等于 m_axi_sg_aclk。 同样，m_axi_sg_aclk 必须小于或等于 m_axi_mm2s_aclk 和 m_axi_s2mm_aclk 中的较慢者。 在同步模式下，所有时钟输入都应连接到相同的时钟信号。 根据连接到 axi_dma 的时钟，在 Vivado IP 集成器中自动设置此参数。

启用分散收集引擎

选中此选项将启用分散收集模式操作，并将分散收集引擎包含在 AXI DMA 中。 取消选中此选项将启用直接寄存器模式操作，但不包括 AXI DMA 中的 Scatter Gather Engine。 禁用 Scatter Gather 引擎会导致 Scatter / Gather 引擎的所有输出端口都绑定为零，并且所有输入端口都保持打开状态。

启用 Micro DMA

选中此选项将生成高度优化的 DMA，而 DMA 资源较少。

此设置可用于传输少量数据的应用程序。 根据所选配置对 DMA 进行编程。 例如，每个事务或每个 BD 可以传输的最大字节数不能超过以下内容：MMap Data_width * Burst_length / 8。

类似地，在此模式下也未实现 4K 边界检查，将寻址范围限制为突发边界。

缓冲区长度寄存器的宽度

此整数值指定用于分散/聚集描述符中传输的控制字段缓冲区长度和状态字段字节的有效位数。 当启用“使用 Rxlength”时，它还会在状态流 App4 的“ RX 长度”字段中指定有效位数。 对于直接寄存器模式，它指定 MM2S_LENGTH 和 S2MM_LENGTH 寄存器中的有效位数。 长度宽度直接与 Scatter / Gather 描述符中指定的字节数或 App4.RxLength，MM2S_LENGTH 或 S2MM_LENGTH 中指定的字节数相关。 字节数等于 2 Length Width。 因此，长度宽度为 26 时得出的字节数为 67,108,863 字节。 对于多通道模式，此值应设置为 23。

地址宽度（32-64）

指定地址空间的宽度。 它可以是 32 到 64 之间的任何值。

启用多通道 DMA

注意：多通道功能即将停用。有关多通道的信息，请参见 AXI 多通道直接内存访问（PG288）[参考资料 12]。

选中此选项将启用 DMA 的多通道功能，并允许您选择 MM2S 和 S2MM 通道的通道数。 有关详细信息，请参见第 2 章中的多通道 DMA 支持。

启用控制/状态流

选中此选项将启用 AXI4 控制和状态流。 AXI4 控制流允许将与 MM2S 通道关联的用户应用程序元数据发送到目标 IP。 MM2S 分散/聚集帧开始（SOF）描述符的用户应用程序字段 0 到 4 在帧 m_axis_mm2s_cntrl 流接口上传输帧开始（TXSOF = 1），并在 m_axis_mm2s 流接口上传输相关的数据包。 AXI4 状态流允许从目标 IP 接收与 S2MM 通道关联的用户应用程序元数据。 接收到的状态数据包将填充 S2MM 分散/聚集帧结束（EOF）描述符的用户应用程序字段 0 至 4。 那是与分组结尾相关联的描述符。 此条件由更新描述符的状态字中的帧接收结束（RXEOF = 1）指示。

启用读取频道选项

以下小节描述了仅影响 AXI 直接内存访问（DMA）内核的 MM2S 通道的选项。

启用频道

此选项启用或禁用 MM2S 通道。 启用 MM2S 通道允许发生从内存到 AXI4-Stream 的读取传输。 禁用 MM2S 通道会将逻辑从 AXI DMA 内核中排除。 MM2S 通道的输出固定为零，AXI DMA 忽略输入。

通道数

此选项指定从 1 到 16 的通道数。

内存映射数据宽度

AXI MM2S 存储器映射读取数据总线的数据宽度（以位为单位）。 有效值为 32、64、128、256、512 和 1,024。

流数据宽度

AXI MM2S AXI4-Stream 数据总线的数据宽度（以位为单位）。 此值必须等于或小于“内存映射数据宽度”。 有效值为 8、16、32、64、128、512 和 1,024。

最大突发尺寸

突发分区粒度设置。此设置指定 MM2S 的 AXI4-Memory Map 一侧的突发周期的最大大小。有效值为 2、4、8、16、32、64、128 和 256。

允许不结盟转移

启用或禁用 MM2S 数据重新排列引擎（DRE）。 选中后，将启用 DRE，并允许将数据重新对齐到 MM2S 内存映射数据路径上的字节（8 位）级别。 对于 MM2S 通道，从内存中读取数据。 如果启用了 DRE，则可以从任何缓冲区地址字节偏移开始读取数据，并且对齐已读取的数据，以使读取的第一个字节是 AXI4-Stream 上的第一个有效字节。

注意：如果为各个通道禁用了 DRE，则不支持未对齐的缓冲区，源或目标地址。 禁用 DRE 的未对齐地址会产生不确定的结果。 DRE 支持仅适用于 512 位及以下的 AXI4-Stream 数据宽度设置。

启用写入通道选项

这些选项仅影响 AXI DMA 内核的 S2MM 通道。

启用通道

此设置启用或禁用 S2MM 通道。 启用 S2MM 通道允许发生从 AXI4-Stream 到内存的写传输。 禁用 S2MM 通道会将逻辑从 AXI DMA 内核中排除。 S2MM 通道的输出固定为零，AXI DMA 忽略输入。

通道数

此选项使您可以选择 1 到 16 个通道。

内存映射数据宽度

AXI S2MM 内存映射写入数据总线的数据宽度（以位为单位）。 有效值为 32、64、128、256、512 和 1,024。

提示：在 Vivado IP 集成器中，此参数是根据流接口的数据宽度自动设置的。 通过将开关更改为“手动”来更新此参数。

流数据宽度

AXI S2MM AXI4-Stream 数据总线的数据宽度（以位为单位）。 此值必须等于或小于“内存映射数据宽度”。 有效值为 8、16、32、64、128、512 和 1,024。

提示：在 Vivado IP 集成器中使用 IP 时，将根据与 s_axis_s2mm 接口建立的连接来自动设置此参数。

最大突发尺寸

此设置指定S2MM通道的AXI4-Memory Map一侧的突发周期的最大大小。换句话说，此设置指定突发映射的粒度。 有效值为2、4、8、16、32、64、128和256。

允许不结盟转移

启用或禁用S2MM数据重新排列引擎（DRE）。 选中后，将启用DRE，并允许将数据重新对齐到S2MM内存映射数据路径上的字节（8位）级别。 对于S2MM通道，数据被写入内存。如果启用了DRE，则数据写入可以从任何缓冲区地址字节偏移开始，并且对齐写入数据，以便将在S2MM AXI4-Stream上接收的第一个有效字节写入指定的未对齐地址偏移。

注意：如果为各个通道禁用了DRE，则不支持未对齐的缓冲区，源或目标地址。 禁用DRE的未对齐地址会产生不确定的结果。 DRE支持仅适用于512位及以下的AXI4-Stream数据宽度设置。

在状态流中使用RxLength

如果启用了控制/状态流，则选中此选项将允许AXI DMA使用状态数据包的App4字段中S2MM目标IP提供的接收长度字段。

这为AXI DMA提供了一个预定的接收字节数，从而使AXI DMA可以命令要传输的确切字节数。

此选项为需要更大吞吐量的系统提供了更高带宽的解决方案。 在此配置中，S2MM目标IP可以提供状态数据包APP4的接收长度字段中指定的所有数据字节。

启用单个AXI4数据接口

仅当在Vivado IP集成器中使用时，此选项才适用。 您可以使用此选项将两个AXI4接口（MM2S和S2MM）组合为一个接口。 此选项不会影响资源或性能。

用户参数

表 4-1 显示了 Vivado IDE 中的字段与用户参数（可以在工具命令语言（Tcl）控制台中查看）之间的关系。

表 4-1：Vivado IDE 参数与用户参数的关系

Vivado IDE Parameter	User Parameter	Default Value
Enable Scatter Gather Engine	c_include_sg	1
Enable Multi Channel Support	c_enable_multi_channel	0
Number of Channels (MM2S)	c_num_mm2s_channels	1
Number of Channels (S2MM)	c_num_s2mm_channels	1
Width of Buffer Length Register	c_sg_length_width	14
Enable Asynchronous clocks	c_prmry_is_aclk_async	0
Enable Control/Status Stream	c_sg_include_stscntrl_strm	1
Enable Micro DMA	c_micro_dma	0
Enable Read Channel	c_include_mm2s	1
Memory Map Data Width (MM2S)	c_m_axi_mm2s_data_width	32
Stream Data Width (MM2S)	c_m_axis_mm2s_tdata_width	32
Allow Unaligned Transfers (MM2S)	c_include_mm2s_dre	0
Max Burst Size (MM2S)	c_mm2s_burst_size	16
Enable Write Channel	c_include_s2mm	1
Use Rxlength in Status Stream	c_sg_use_stsapp_length	0
Memory Map Data Width (S2MM)	c_m_axi_s2mm_data_width	32
Stream Data Width (S2MM)	c_s_axis_s2mm_tdata_width	32
Allow Unaligned Transfers (S2MM)	c_include_s2mm_dre	0
Max Burst Size (S2MM)	c_s2mm_burst_size	16
Address Width (32-64)	c_addr_width	32

输出产生

有关详细信息，请参见《 Vivado Design Suite 用户指南：使用 IP 设计（UG896）[参考 1]》。

- 约束核心

Vivado 设计套件中随核心生成一起提供了必要的 XDC 约束。

- 必需的约束

本节不适用于此 IP 内核。

- 设备，封装和速度等级的选择

本部分不适用于此 IP 内核。

- 时钟频率

本部分不适用于此 IP 内核。

- 时钟管理

本部分不适用于此 IP 内核。

- 时钟放置

本节不适用于此 IP 内核。

- Banking

本部分不适用于此 IP 内核。

- 收发器放置

本节不适用于此 IP 内核。

- I / O 标准和位置

本节不适用于此 IP 内核。

模拟

有关 Vivado 仿真组件的全面信息以及有关使用受支持的第三方工具的信息，请参见《 Vivado Design Suite 用户指南：逻辑仿真（UG900）[参考 6]》。

重要信息：对于面向 7 系列或 Zynq-7000 设备的内核，不支持 UNIFAST 库。Xilinx IP 仅通过 UNISIM 库进行测试和认证。

综合与实施

有关综合和实现的详细信息，请参见《 Vivado Design Suite 用户指南：使用 IP 进行设计（UG896）[参考 1]。

样例设计

本章包含有关 Vivado®Design Suite 中提供的示例设计的信息。

顶层模块实例化了实现硬件设计所需的核心和示例设计的所有组件，如图 5-1 所示。这包括混合模式时钟管理器（MMCME2），寄存器配置，数据生成器和数据检查器模块。

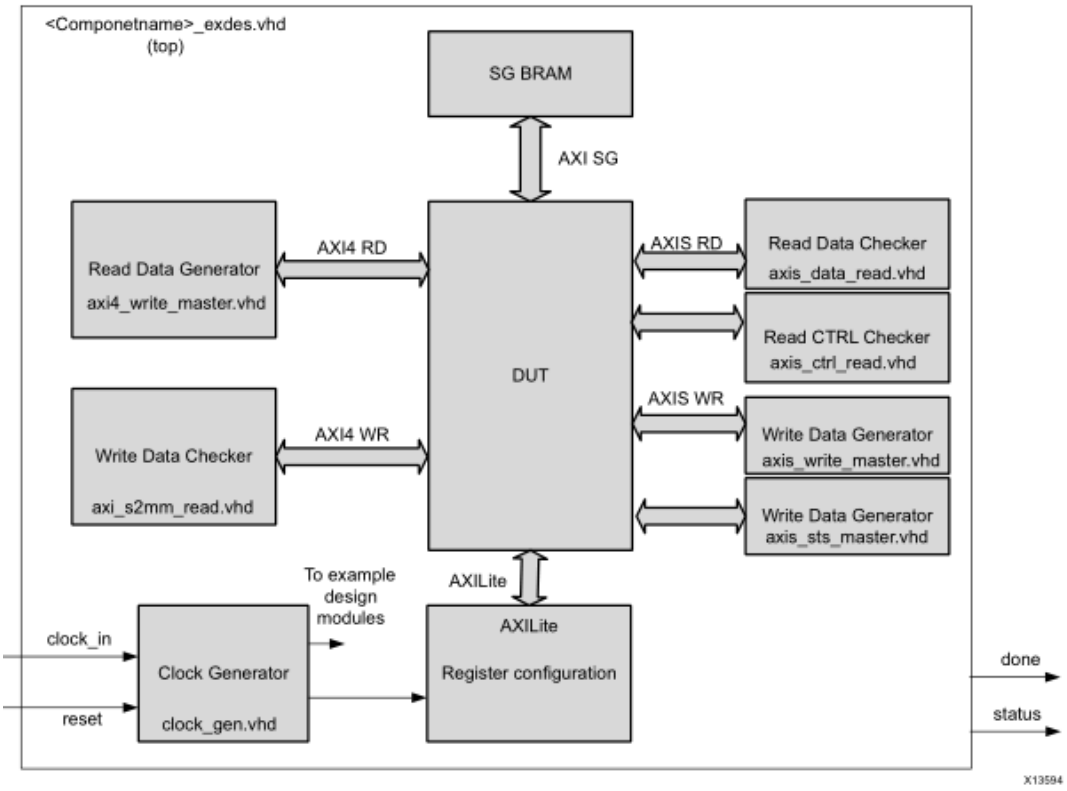

Figure 5-1: **Block Diagram of Example Design**

此示例设计演示了 DUT 的 AXI4-Lite，AXI4 和 AXI4-Stream 接口上的事务。

时钟发生器：MMCME2 用于为示例设计生成时钟。 当 DUT 处于同步模式时，MMCME2 在示例设计中为所有 AXI 接口生成 100 MHz 时钟。 在异步模式下，MMCME2 为 AXI4-Lite 接口生成 50 MHz 时钟，为 AXI4 和 AXI4-Stream 接口生成 100 MHz 时钟。

示例设计中的 DUT 和其他模块将保持复位状态，直到 MMCME2 被锁定。

寄存器配置模块：该模块配置 DUT 寄存器，如第 3 章“编程顺序”中的编程顺序所述。该模块是 AXI Traffic Generator 模块，用于对寄存器进行编程。 有关更多信息，请参考 AXI Traffic Generator LogiCORE IP（PG125）[参考 7]。

读取路径生成器：它使用 AXI Block RAM，在锁定 MMCME2 之后将其填充（具有固定的传输量）。 MM2S 通道读取此 AXI Block RAM 并将数据传输到 AXI4-Stream 接口。

读取路径检查器：该模块检查在 MM2S AXI4-Stream 接口上传输的数据。

读取路径 CTRL 检查器：该模块检查 MM2S AXI4-Stream 控制接口上传输的数据。

写路径生成器：配置写（S2MM）通道后，此模块在 S2MM AXI4-Stream 接口上驱动事务（具有固定的传输量）。

写入路径 STS 生成器：此模块生成 S2MM STS 数据流。

写入路径检查器：该模块检查在 AXI4 接口上接收的数据。 在 AXI4 接口上接收的数据也被写入另一个 AXI Block RAM。

MMCME2 锁定后，测试将立即开始。 完成所有事务后，将完成引脚置为高电平。 类似地，当数据完整性检查成功时，状态引脚被置为高电平。 这两个引脚可以连接到 LED，以了解测试状态。

实施示例设计

在遵循了第 4 章“设计流程步骤”中所述的步骤之后，按如下所示实施示例设计：

1.在“层次结构”窗口中右键单击核心，然后选择“打开 IP 示例设计”。

2.弹出一个新窗口，要求您为示例设计指定目录。 选择一个新目录，或保留默认目录。

将在所选目录中自动创建一个新项目，并在新的 Vivado IDE 窗口中打开。

3.在 Flow Navigator（左侧窗格）中，单击 Run Implementation，然后按照说明进行操作。

表 5-1，表 5-2 和表 5-3 描述了示例设计，仿真和约束目录中的文件。

在当前项目目录中，将创建一个名为<component_name> _example 的新项目，并将文件发送到该目录中。 该目录及其子目录包含创建 AXI DMA 控制器示例设计所需的所有源文件。

表 5-1 显示了示例设计中的文件。

表 5-1：示例设计目录

名称	描述
<component_name>_exdes.vhd	示例设计的顶级 HDL 文件。
axi_lite_sm.vhd	注册配置文件以进行示例设计。 （默认情况下不使用此文件。您可以更新<component_name> _exdes.vhd 以代替 AXI Traffic Generator 使用）。
clock_gen.vhd	时钟生成模块为例设计。
axi4_write_master.vhd	读取路径数据生成器模块的示例设计。
axis_data_read.vhd	读取路径数据检查器模块以进行示例设计。
axis_write_master.vhd	写路径数据生成器模块的示例设计。
axi_s2mm_read.vhd	编写路径数据检查器模块以进行示例设计。
axis_ctrl_read.vhd	MM2S CTRL 数据检查器
axis_sts_master.vhd	S2MM STS 数据生成器
sg_mif.coe	块存储器用于存储 BD 的 COE 文件

表 5-2 显示了可用于运行模拟的测试平台文件。

表 5-2：仿真目录

名称	描述
<component_name>_exdes_tb.vhd	示例设计的测试台

表 5-3 显示了实现示例设计所需的 XDC 文件。
表 5-3：约束目录

名称	描述
<component_name>_exdes.xdc	示例设计的顶级约束文件。

示例设计提供的 XDC 具有为 KC705 板配置的所有 I / O 引脚。
这些约束默认为注释。 在继续实施 KC705 板之前，请先取消注释。

模拟示例设计

使用 AXI DMA 示例设计（作为 AXI DMA 的一部分提供），可以快速模拟和观察 AXI DMA 的行为。

仿真脚本编译了 AXI DMA 示例设计和支持的仿真文件。 然后，它运行模拟并检查以确保成功完成。

如果测试失败，则会显示以下消息：测试失败!!!

如果测试通过，则显示以下消息：测试成功完成
如果测试挂起，则显示以下消息：测试失败！ 测试超时

示例设计测试台

本节包含有关 Vivado Design Suite 中提供的测试平台的信息。

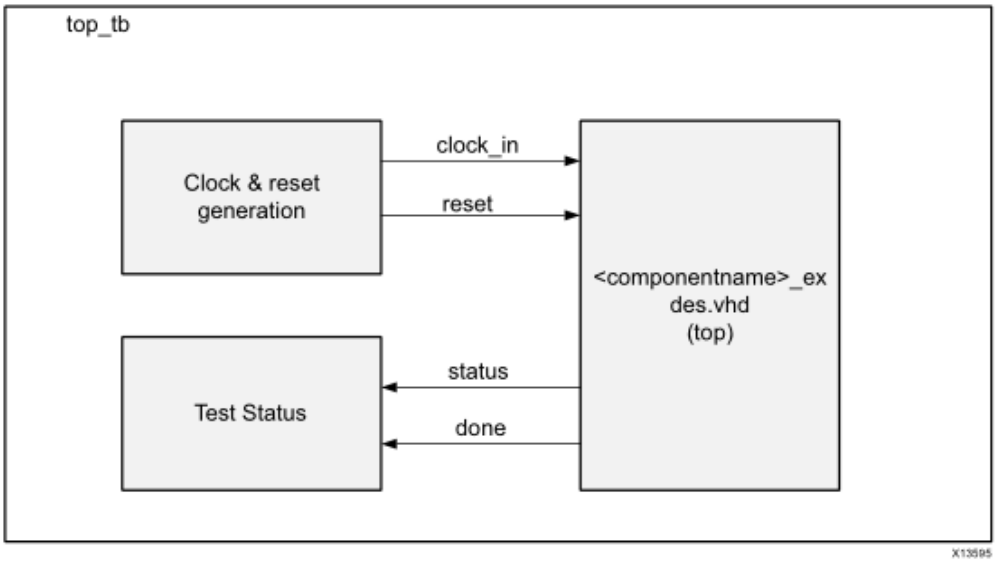

Figure 5-2: **AXI DMA Example Design Test Bench**

图 5-2 显示了 AXI DMA 示例设计的测试平台。顶层测试台产生一个 200 MHz 的差分时钟，并驱动对示例设计的初始复位。

升级中

本附录包含有关将设计从 ISE®迁移到 Vivado®Design Suite，以及升级到 IP 内核的最新版本的信息。 对于在 Vivado Design Suite 中进行升级的客户，其中包括有关任何端口更改以及对用户逻辑的其他影响的重要详细信息（如果适用）。

迁移到 Vivado 设计套件

有关从 ISE 工具迁移到 Vivado Design Suite 的信息，请参阅 ISE 到 Vivado Design Suite 迁移指南（UG911）[参考 8]。

在 Vivado Design Suite 中升级

本节提供有关在 Vivado Design Suite 中升级到此 IP 内核的最新版本时对用户逻辑或端口名称所做的任何更改的信息。 该内核没有任何参数或端口更改。

调试

本附录包括有关 Xilinx 支持网站和调试工具上可用资源的详细信息。

- 在 Xilinx.com 上查找帮助

为了在使用 AXI DMA 内核时为设计和调试过程提供帮助，Xilinx 支持网页包含关键资源，例如产品文档，发行说明，答复记录，有关已知问题的信息以及用于获取更多信息的链接。 产品支持。

- 文档

本产品指南是与 AXI DMA 内核相关的主要文档。 该指南以及与所有有助于设计过程的产品有关的文档，可以在 Xilinx 支持网页上找到，也可以使用 Xilinx®Documentation Navigator 找到。

从下载页面下载 Xilinx 文档浏览器。 有关此工具和可用功能的更多信息，请在安装后打开联机帮助。

- 答案记录

答案记录包括有关常见问题的信息，有关如何解决这些问题的有用信息以及 Xilinx 产品的任何已知问题。

每天都会创建和维护答案记录，以确保用户可以访问最准确的信息。

可以通过使用 Xilinx 支持主页上的“搜索支持”框找到该核心的答案记录。为了使搜索结果最大化，请使用适当的关键字，例如
•产品名称
•工具消息
•遇到的问题摘
要返回结果后，可以进行过滤器搜索以进一步确定结果。

AXI DMA 核心的主答复记录

Xilinx 答复 54682

- 技术支持

如产品文档中所述使用时，Xilinx 在此 LogiCORE™IP 产品的 Xilinx 支持网页上提供技术支持。 如果您执行以下任何一项操作，Xilinx 不能保证时间，功能或支持：

•在文档中未定义的设备中实施解决方案。

•自定义解决方案，超出产品文档中允许的范围。

•更改设计中标有“请勿修改”的任何部分。

要联系 Xilinx 技术支持，请导航至 Xilinx 支持网页。

Vivado Design Suite 调试功能

Vivado®Design Suite 调试功能可将逻辑分析仪和虚拟 I / O 内核直接插入设计中。 调试功能还允许您设置触发条件，以捕获硬件中的应用程序和集成块端口信号。 然后可以分析捕获的信号。 Vivado IDE 中的此功能用于逻辑调试和 Xilinx 器件中运行的设计的验证。

Vivado 逻辑分析仪与逻辑调试 IP 内核一起使用，包括：

•ILA 2.0（及更高版本）

•VIO 2.0（及更高版本）

几个内部信号被标记为调试信号。 这些可以轻松添加到逻辑分析仪中。

请参见《 Vivado 设计套件用户指南：编程和调试（UG908）[参考 9]》。

硬件调试

硬件问题的范围可以从链接启动到经过数小时的测试后看到的问题。 本节提供了常见问题的调试步骤。

随后会遇到一些常见问题和可能的解决方案。

•您已对 BD 环进行了编程，但似乎无作用。 必须遵循寄存器编程顺序才能启动 DMA。 请参阅编程顺序和描述符管理。
•状态寄存器中设置的内部错误/错误位
    °当描述符中指定的 BTT 为 0 时，将设置内部错误
    ° 如果获取的 BD 是完整的 BD，则会设置 SG 内部错误。
    ° 其他错误位，例如解码错误或从属错误，也将根据来自互连或从属的响应进行设置。
•您正在从某个位置读取数据，但是数据似乎不整齐。

验证起始地址位置是否对齐。 如果未对齐，请确保在配置 DMA 时启用了 DRE。

另请参阅 Xilinx 解决方案中心，以在设计周期的所有阶段获得对设备，软件工具和知识产权的支持。 主题包括设计帮助，咨询和故障排除技巧。

其他资源和法律声明

Xilinx 资源

有关答案，文档，下载和论坛等支持资源，请参见 Xilinx 支持。

文档导航器和设计中心

Xilinx 文档导航器提供对 Xilinx 文档，视频和支持资源的访问，您可以对其进行过滤和搜索以查找信息。 要打开 Xilinx 文档浏览器（DocNav）：
•从 Vivado®IDE 中，选择“帮助”>“文档和教程”。

•在 Windows 上，选择开始>所有程序> Xilinx 设计工具> DocNav。

•在 Linux 命令提示符下，输入 docnav。

Xilinx 设计中心提供了按设计任务和其他主题组织的文档的链接，您可以使用它们来学习关键概念并解决常见问题。 要访问设计中心：
•在 Xilinx 文档导航器中，单击“设计中心视图”选项卡。

•在 Xilinx 网站上，请参见“设计中心”页面。

注：有关 Documentation Navigator 的更多信息，请参见 Xilinx 网站上的 Documentation Navigator 页面。

参考

要搜索 Xilinx 文档，请访问 www.xilinx.com/cn。

1. Vivado Design Suite User Guide: Designing with IP (UG896)
2. AXI Reference Guide (UG1037)
3. AMBA AXI and ACE Protocol Specification (ARM IHI 0022D)
4. Vivado Design Suite User Guide: Getting Started (UG910)
5. Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator (UG994)
6. Vivado Design Suite User Guide - Logic Simulation (UG900)
7. AXI Traffic LogiCORE IP Generator (PG125)
8. ISE to Vivado Design Suite Migration Guide (UG911)
9. Vivado Design Suite User Guide: Programming and Debugging (UG908)
10. AXI Interconnect LogiCORE IP Product Guide (PG059)
11. Synthesis and Simulation Design Guide (UG626)
12. AXI Multichannel Direct Memory Access (PG288)

请阅读：重要法律声明

以下提供给您的信息（“材料”）仅用于 Xilinx 产品的选择和使用。 在适用法律允许的最大范围内：（1）材料按“原样”提供，且有所有错误，Xilinx 特此声明，不作任何明示，暗示或法定的担保和条件，包括但不限于对适销性的担保，非 -侵权或适合任何特定目的；（2）对于与材料相关，由材料引起或与材料相关的任何种类或性质的任何损失或损害，Xilinx 不承担任何责任（无论是合同还是侵权，包括过失，或根据任何其他责任理论） （包括您对资料的使用），包括任何直接，间接，特殊，偶然或结果性的损失或损害（包括数据损失，利润，商誉或因采取任何行动而遭受的任何类型的损失或损害） 第三方)），即使此类损害或损失是可以合理预见的，或者已告知 Xilinx 也可能这样做。 Xilinx 不承担更正材料中包含的任何错误或将材料或产品规格的更新通知您的义务。未经事先书面许可，您不得复制，修改，分发或公开展示材料。 某些产品受 Xilinx 有限保修条款的约束，请参阅 Xilinx 的销售条款，该条款可在 https://www.xilinx.com/legal.htm#tos 上查看。 IP 内核可能受 Xilinx 颁发给您的许可中包含的保修和支持条款的约束。 Xilinx 产品并非旨在或旨在提供故障保护功能，也不旨在用于需要故障保护性能的任何应用中； 您承担在此类关键应用中使用 Xilinx 产品的唯一风险和责任，请参阅 Xilinx 的销售条款，该条款可在 https://www.xilinx.com/legal.htm#tos 上查看。

- 汽车应用免责声明

除非存在安全概念或冗余特征，否则不保证将汽车产品（在零件号中标识为“ XA"）用于部署气囊或用于影响车辆控制的应用程序（“安全应用程序”）。 符合 ISO 26262 汽车安全标准（“安全设计"）。 在使用或分发包含产品的任何系统之前，客户应彻底测试此类用于安全目的的系统。 在没有安全设计的情况下，在安全应用中使用产品完全由客户承担风险，但前提是要遵守适用于产品责任限制的法律和法规。