

# DMA/Bridge Subsystem for PCI Express v4.1

## 产品指南

Vivado Design Suite

PG195 (v4.1) 2021 年 4 月 29 日

条款中英文版本如有歧义，概以英文版本为准。



# 目录

第 1 章：简介.....	4
功能特性.....	4
IP 相关信息.....	5
第 2 章：概述.....	6
特性总结.....	7
应用.....	8
不支持的功能.....	8
限制.....	8
许可和订购.....	10
第 3 章：产品规格.....	11
标准.....	11
性能和资源使用情况.....	11
器件最低要求.....	11
子系统的可配置组件.....	12
DMA 操作.....	16
端口描述.....	25
寄存器空间.....	37
第 4 章：利用子系统进行设计.....	69
时钟设置和复位.....	69
串联配置.....	69
第 5 章：设计流程步骤.....	74
自定义和生成子系统.....	74
约束子系统.....	83
仿真.....	85
综合与实现.....	87
第 6 章：设计示例.....	88
可用设计示例.....	88
自定义和生成设计示例.....	98
第 7 章：测试激励文件.....	99
对应端点的根端口模型测试激励文件.....	99
附录 A：应用软件开发.....	106

器件驱动程序.....	106
Linux 器件驱动程序.....	106
使用驱动程序.....	107
中断处理.....	107
H2C 流程示例.....	108
C2H 流程示例.....	108
 附录 B：升级.....	 109
新参数.....	109
新增端口.....	109
 附录 C：调试.....	 112
在 Xilinx.com 上寻求帮助.....	112
调试工具.....	113
硬件调试.....	114
 附录 D：使用赛灵思虚拟线缆进行调试.....	 115
简介.....	115
主机 PC XVC-Server 应用.....	116
主机 PC XVC-over-PCIe 驱动.....	116
启用 XVC-over-PCIe 的 FPGA 设计.....	116
使用 PCIe-XVC-VSEC 设计示例.....	121
 附录 E：附加资源与法律提示.....	 129
赛灵思资源.....	129
Documentation Navigator 与设计中心.....	129
参考资料.....	129
修订历史.....	130
请阅读：重要法律提示.....	133

## 简介

赛灵思 DMA/Bridge Subsystem for PCI Express® (PCIe®) 可实现高性能的可配置分散聚集 (Scatter Gather) DMA 以配合 PCI Express® 2.1 和 3.x 集成块 (Integrated Block) 一起使用。此 IP 支持选择 AXI4 Memory Mapped 或 AXI4-Stream 用户接口。

此 IP 还支持可选 PCIe AXI Bridge 模式，此模式仅对 UltraScale+™ 器件启用。如需了解有关 PCIe AXI Bridge 模式操作的详细信息，请参阅《AXI Bridge for PCI Express Gen3 Subsystem 产品指南》(PG194)。

本文档仅涵盖 DMA 模式操作。

**注释：**如需了解有关 Versal ACAP 子系统的详细信息，请参阅《Versal ACAP DMA and Bridge Subsystem for PCI Express 产品指南》(PG344)。

---

## 功能特性

- 支持 UltraScale+™、UltraScale™、Virtex®-7 XT Gen3（端点）和 7 系列 2.1（端点）Integrated Blocks for PCIe。不支持 7A15T 和 7A25T
- 支持 64、128、256、512 位数据路径（针对 7 系列 Gen2 IP 仅支持 64 和 128 位数据路径）
- 64 位源地址、目标地址和描述符地址
- 最多 4 条主机到卡（H2C/读取）数据通道（针对 7 系列 Gen2 IP 最多 2 条数据通道）
- 最多 4 条卡到主机（C2H/写入）数据通道（针对 7 系列 Gen2 IP 最多 2 条数据通道）
- 可选用户接口
  - 单一 AXI4 Memory Mapped (MM) 用户接口
  - AXI4-Stream 用户接口（每条通道都有其自己的 AXI4-Stream 接口）
- AXI4 主接口和 AXI4-Lite 主接口（可选）允许 PCIe 流量绕过 DMA 引擎
- AXI4-Lite 从接口，用于访问 DMA 状态寄存器
- 分散聚集描述符列表，支持无限列表大小
- 每个描述符最大传输大小为 256 MB
- 传统中断、MSI 中断和 MSI-X 中断
- 块提取连续描述符
- 轮询模式
- 描述符旁路接口
- 任意源地址和目标地址
- 在 AXI 总线上进行奇偶校验检查或传输奇偶校验（对于 7 系列 Gen2 IP 不可用）

## IP 相关信息

LogiCORE™ IP 相关信息表	
子系统规格	
支持的器件系列 <sup>1</sup>	UltraScale+ 器件、UltraScale 器件和 7 系列 Gen2 器件
支持的用户接口	AXI4 MM、AXI4-Lite 和 AXI4-Stream
资源	请参阅 <a href="#">资源使用情况网页</a> 。
随子系统提供	
设计文件	加密 System Verilog
设计示例	Verilog
测试激励文件	Verilog
约束文件	XDC
仿真模型	Verilog
支持的软件驱动程序	Linux 和 Windows 驱动程序 <sup>2</sup>
经过测试的设计流程 <sup>3</sup>	
设计输入	Vivado® Design Suite
仿真	如需了解受支持的仿真器，请参阅 <a href="#">赛灵思设计工具：版本说明指南</a> 。
综合	Vivado 综合
支持	
版本说明和已知问题	主答复记录： <a href="#">AR 65443</a>
所有 Vivado IP 变更日志	Vivado IP 主变更日志： <a href="#">72775</a>
<a href="#">赛灵思支持网页</a>	

### 注释：

1. 有关所支持器件的完整列表，请参阅 Vivado® IP 目录。
2. 欲知详情，请参阅 [附录 A: 应用软件开发](#) 和 [答复记录 65444](#)。
3. 如需了解受支持的工具版本，请参阅[赛灵思设计工具：版本说明指南](#)。
4. 对于 Versal ACAP，请参阅《Versal ACAP DMA and Bridge Subsystem for PCI Express 产品指南》([PG344](#))。

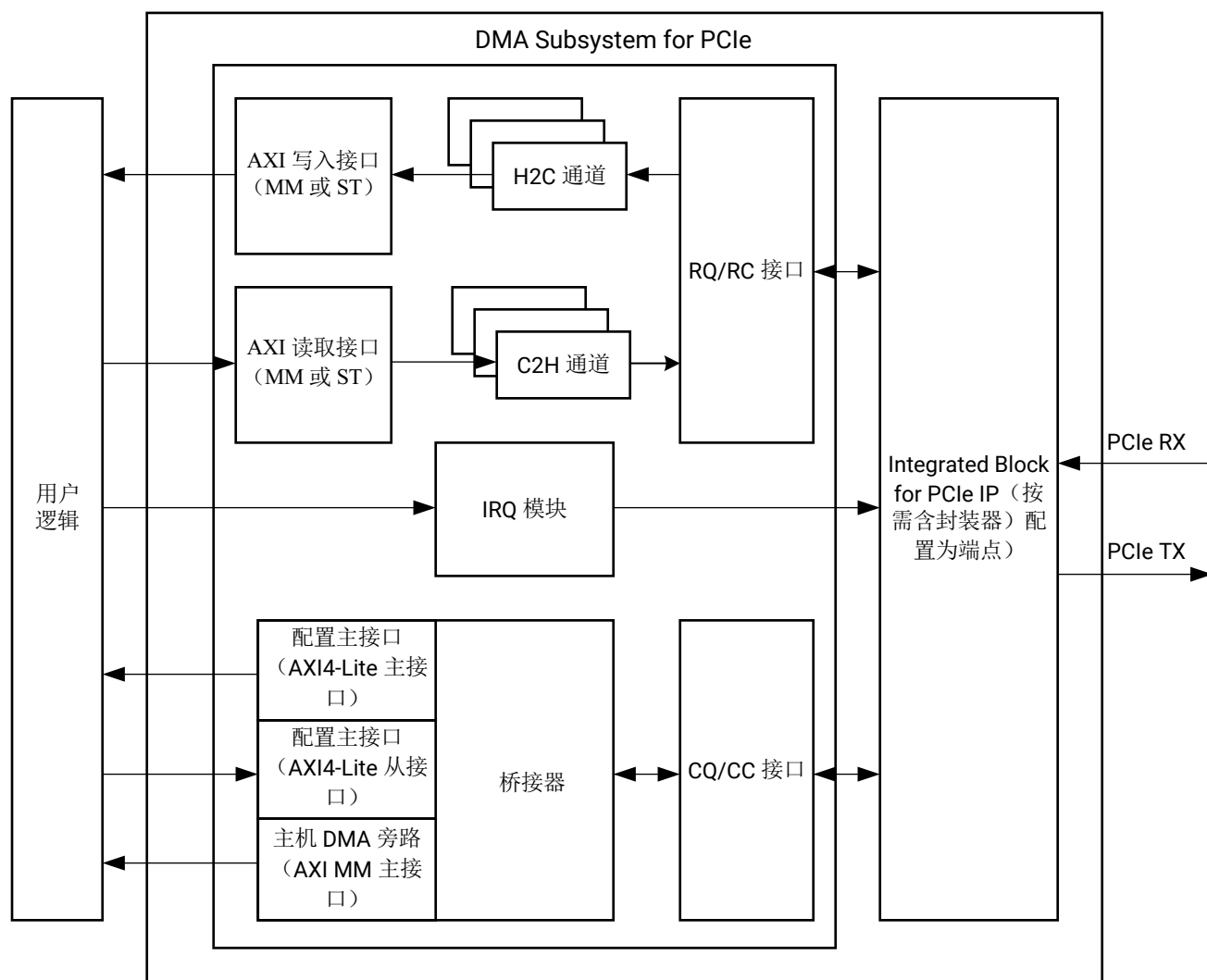
# 概述

DMA/Bridge Subsystem for PCI Express<sup>®</sup> (PCIe<sup>®</sup>) 可配置为 PCI Express 与 AXI 存储器空间之间的高性能直接存储器访问 (DMA) 数据移动器或桥接器。

- DMA 数据移动器：作为 DMA，该核可通过 AXI Memory Mapped 接口或者通过 AXI Streaming 接口来配置，以支持直接连接至 RTL 逻辑。在 PCIe 地址空间与 AXI 地址空间之间可使用所提供的字符驱动程序通过以上任一接口进行高性能块数据移动。除了基本 DMA 功能外，DMA 还支持最多 4 条上游和下游通道、支持 PCIe 流量绕过 DMA 引擎（主机 DMA 旁路），还支持通过可选描述符旁路来管理来自 FPGA 互连结构的描述符，以满足需要最高性能和最低时延的应用的需求。
- PCIe 与 AXI 存储器之间的桥接器：配置为 PCIe Bridge 时，接收到的 PCIe 数据包将被转换为 AXI 流量，接收到的 AXI 流量则将被转换为 PCIe 流量。桥接功能适合需要快速轻松访问 PCI Express 子系统的 AXI 外设使用。桥接功能可用作端点 (Endpoint) 或根端口 (Root Port)。仅限 UltraScale+<sup>™</sup> 器件才支持 PCIe Bridge 功能。非 UltraScale+ 器件具有专用的 PCIe Bridge IP，可在 Vivado<sup>®</sup> IP 目录中找到。如需了解有关 PCIe Bridge 模式操作的详细信息，请参阅《AXI Bridge for PCI Express Gen3 Subsystem 产品指南》(PG194)。

本文档仅涵盖 DMA 模式操作。

图 1：DMA/Bridge Subsystem for PCI Express® 概述



X14718-051621

此图展示了请求方请求 (RQ) 接口、请求方完成 (RC) 接口、完成方请求 (CQ) 接口和完成方完成 (CC) 接口。如需了解有关这些接口的更多信息，请参阅《UltraScale+ Integrated Block for PCI Express LogiCORE IP 产品指南》(PG213)。

## 特性总结

DMA/Bridge Subsystem for PCI Express® 支持在主机存储器与 DMA 子系统之间移动数据。具体方式是对包含有关源、目标以及要传输的数据量的信息的“描述符”进行操作。这些直接存储器传输可在主机到卡 (H2C) 和卡到主机 (C2H) 传输中执行。DMA 可配置为包含单个 AXI4 主接口（供所有通道共享），或针对启用的每条通道包含 1 个 AXI4-Stream 接口。存储器传输在已链接的描述符列表中按通道来执行，DMA 将从该列表中提取主机存储器和进程。诸如描述符完成和错误等事件均使用中断来发出信号。该核还提供最多 16 个用户中断连线，用于向主机生成中断。

主机能够通过以下 2 个接口直接访问用户逻辑：

- AXI4-Lite 主接口配置端口：此端口为固定 32 位端口，用于对用户配置和状态寄存器执行非性能关键性访问。
- AXI Memory Mapped 主接口 CQ 旁路端口：此端口的宽度与 DMA 通道数据路径相同，用于对用户存储器执行高带宽访问，可满足诸如对等传输等应用的需求。

用户逻辑能够通过 AXI4-Lite 从配置接口访问 DMA/Bridge Subsystem for PCIe 内部配置和状态寄存器。在此接口上主控的请求将不会被转发至 PCI Express。

---

## 应用

该核架构支持各种计算和通信目标应用，并强调性能、成本、可缩放性、功能可扩展性以及对于任务至关重要的可靠性等因素。典型应用包括：

- 数据通信网络
- 远程通信网络
- 宽带有线和无线应用
- 网络接口卡
- 芯片到芯片 (Chip-to-Chip) 和背板接口卡
- 适用于各种应用的服务器插卡

---

## 不支持的功能

在此核中，不支持以下标准功能：

- 针对 Virtex®-7 XT 和 7 系列 Gen2 器件不支持串联配置解决方案（串联 PROM、串联 PCIe、含现场更新的串联、PR over PCIe）
- 当前在 UltraScale+ 器件中对于 Bridge 模式暂不支持串联配置。
- SR-IOV
- ECRC
- 并非所有配置都支持设计示例
- 窄突发（在主接口上不受支持）
- BAR 转换，用于将 DMA 地址转换为 AXI4 Memory Mapped 接口

---

## 限制

### PCIe 传输事务类型

可生成的 PCIe® 传输事务均为符合 AXI4 规范的传输事务。下表列出了受支持的 PCIe 传输事务类型。



表 1：受支持的 PCIe 传输事务类型

TX	RX
MRd32	MRd32
MRd64	MRd64
MWr32	MWr32
MWr64	MWr64
Msg(INT/Error)	Msg(SSPL,INT,Error)
Cpl	Cpl
CplD	CplD

## PCIe 功能

对于 DMA Subsystem for PCIe®, 根据 AXI4 规格, 仅支持下列 PCIe 功能:

- 1 PF
- MSI
- MSI-X
- PM
- AER (仅限 PCIe 3.x 核)

## 其它

- 仅支持 INCR 突发类型。其它类型会导致从接口非法突发 (SIB) 中断。
- 不支持存储器类型 (AxCACHE)
- 不支持保护类型 (AxPROT)
- 不支持锁定类型 (AxLOCK)
- 不支持非连续字节使能 (WSTRB)
- 对于 7 系列 Gen2 IP, 从主机系统执行的 PCIe 访问必须限制为 1DW (4 个字节) 传输事务。

**注释:** AXI 旁路和寄存器访问均适用此限制。

## PCIe 到 DMA 旁路主接口

- 仅发出 INCR 突发类型
- 仅发出非特权保护类型的非安全数据。
- 对于 7 系列 Gen2 IP, 仅限于 1DW (4 个字节) 传输事务。

## MSI-X 模式下的用户中断

用户需在“IRQ Block User Vector Number”寄存器中通过编程来为每个用户中断提供不同的矢量编号, 以便为所有用户中断生成 acks。这样即可在存在同步中断时为所有用户中断生成 acks。当所有矢量编号指向同一个 MSI-X 条目时, 只有 1 个 ack。

---

## 许可和订购

根据[赛灵思最终用户许可条款](#)，此赛灵思 IP 模块随附赛灵思 Vivado<sup>®</sup> Design Suite 免费提供。如需获取有关此模块及其它赛灵思 IP 模块的信息，请访问[赛灵思知识产权](#)页面。如需获取有关其它赛灵思 IP 模块和工具的定价和可用性信息，请联系您[当地的赛灵思销售代表](#)。

如需了解更多信息，请访问 [DMA Subsystem for PCI Express](#) 产品页面。

# 产品规格

DMA/Bridge Subsystem for PCI Express® (PCIe®) 配合 Integrated Block for PCI Express IP 一起使用即可提供高度可配置的 DMA Subsystem for PCIe 和高性能 DMA 解决方案。

## 标准

DMA/Bridge Subsystem for PCIe 符合以下规范：

- 《AMBA AXI4-Stream 协议规范》([ARM IHI 0051A](#))
- [PCI Express 基本规范 v2.1 和 v3.0](#)

## 性能和资源使用情况

有关 DMA 性能数据的信息，请参阅[赛灵思答复记录 68049](#)。

有关 DMA 资源使用情况的信息，请参阅[资源使用情况网页](#)。

## 器件最低要求

下表列出了给定速度等级的链路宽度和受支持的速度。

表 2：器件最低要求

功能链路速度	功能链路宽度	受支持的速度等级
UltraScale+™ 架构 (PCIe4)		
Gen1/Gen2	x1、x2、x4、x8 和 x16	-1、-1L、-1LV、-2、-2L、-2LV 和 -3 <sup>1</sup>
Gen3	x1、x2 和 x4	-1、-1L、-1LV、-2、-2L、-2LV 和 -3 <sup>1</sup>
	x8	-1、-2、-2L、-2LV 和 -3 <sup>1</sup>
	x16	-1、-2、-2L 和 -3 <sup>1</sup>
含 HBM 的 Virtex® UltraScale+™ 器件 (PCIe4C) <sup>2</sup>		
Gen1/Gen2	x1、x2、x4、x8 和 x16	-1、-2、-2L、-2LV 和 -3

表 2：器件最低要求 (续)

功能链路速度	功能链路宽度	受支持的速度等级
Gen3	x1、x2 和 x4	-1、-2、-2L、-2LV 和 -3
	x8	-1、-2、-2L、-2LV 和 -3
	x16	-1、-2、-2L、-2LV 和 -3
Gen4 <sup>6</sup>	x1、x2、x4 和 x8	-2、-2L 和 -3
<b>UltraScale™ 器件</b>		
Gen1	x1、x2、x4 和 x8	-1、-1L、-1LV、-1H、-1HV、-2 和 -3 <sup>3</sup>
Gen2	x1、x2、x4 和 x8	-1、-1L、-1LV、-1H、-1HV、-2 和 -3 <sup>3</sup>
Gen3	x1、x2 和 x4	-1、-1L、-1LV、-1H、-1HV、-2 和 -3 <sup>3,4</sup>
Gen3	x8	-2 和 -3
<b>7 系列 Gen3 器件</b>		
Gen1	x1、x2、x4 和 x8	-1、-1M、-1I、-2、-2L、-2G、-2I 和 -3
Gen2	x1、x2、x4 和 x8	-1、-1M、-1I、-2、-2L、-2G、-2I 和 -3
Gen3	x1、x2、x4 和 x8	-2、-2L、-2G、-2I 和 -3
<b>7 系列 Gen2 器件</b>		
Gen1	x1、x2、x4 和 x8	-1 <sup>5</sup> 、-2 <sup>5</sup> 和 -3
Gen2	x1、x2 和 x4	-1 <sup>5</sup> 、-2 <sup>5</sup> 和 -3
	x8	-2 <sup>5</sup> 和 -3

**注释：**

1. -1L(0.95V)、-1LV(0.90V)、-2L(0.85V) 和 -2LV(0.72V)。
2. 含高带宽存储器 (HBM) 的 Virtex® UltraScale+™ 器件包含 PCIe 块和 PCIe4C 块。仅限 PCIe4C 块才支持 -2LV 速度等级的 Gen3 x16。
3. -1L(0.95V)、-1LV(0.90V)、-1H(1.0V) 和 -1HV(0.95V)。
4. “核时钟频率 (Core Clock Frequency)” 选项针对 -1、-1LV、-1L、-1H 和 -1HV 速度等级必须设置为 250 MHz。
5. 根据所选系列，可用 -1 速度等级包括 -1M、-1I 和 -1Q。根据所选系列，可用 -2 速度等级包括 -2、-2G、-2I、-2IL 和 -2L。
6. 如需了解 Gen4 模式限制，请参阅《UltraScale+ Integrated Block for PCI Express LogiCORE IP 产品指南》(PG213)。

## 子系统的可配置组件

子系统内部可配置为实现最多 8 个独立物理 DMA 引擎（针对 H2C 和 C2H 最多各 4 个）。这些 DMA 引擎可映射到用户应用的独立 AXI4-Stream 接口或共享 AXI4 Memory Mapped (MM) 接口。在 AXI4 MM 接口上，DMA/Bridge Subsystem for PCI Express® 可生成请求和期望的完成包。AXI4-Stream 接口为仅含数据的接口。

配置的通道类型可用于判定发生传输事务的总线。

- 主机到卡 (H2C) 通道可向 PCIe 生成读取请求并向用户应用提供数据或者生成写入请求。
- 同样，卡到主机 (C2H) 通道会等待用户侧的数据，或者在用户侧生成读取请求，然后向 PCIe 生成包含接收到的数据的写入请求。

DMA/Bridge Subsystem for PCIe 还支持主机访问用户逻辑。到达“PCIe 到 DMA 旁路基址寄存器 (BAR)”的写入请求由 DMA 进行处理。来自该写入请求的数据将通过 M\_AXI\_BYPASS 接口转发至用户应用。

主机访问用户逻辑中的配置和状态寄存器的能力是通过 AXI4-Lite 主端口提供的。这些请求均为 32 位读取或写入。用户应用也可通过 AXI4-Lite 从端口访问内部 DMA 配置和状态寄存器。

为 H2C 和 C2H 启用多个通道时，AXI4 主接口上的传输事务将在所有选中的通道内进行交织。在此情况下会使用简单的循环协议来维护所有通道。传输事务的粒度取决于主机最大有效载荷大小 (MPS)、页面大小和其它主机设置。

## 目标桥接器

目标桥接器用于接收来自主机的请求。根据 BAR，请求将通过 AXI4-Lite 主接口转发至内部目标用户或 CQ 旁路端口。在下游用户逻辑返回非转发请求的数据后，目标桥接器就会生成读取完成 TLP，并通过 CC 总线将其发送至 PCIe IP。

在下表中，PCIe BAR 选择对应于 Vivado® 集成设计环境 (IDE) 的“PCIe BARs”选项卡下设置的选项。

表 3：32 位 BAR

IP 自定义期间的 PCIe BAR 选择	BAR0 (32 位)	BAR1 (32 位)	BAR2 (32 位)
默认值	DMA		
启用“PCIe to AXI Lite Master”	PCIe 到 AXI4-Lite 主接口	DMA	
启用“PCIe to AXI Lite Master”和“PCIe to DMA Bypass”	PCIe 到 AXI4-Lite 主接口	DMA	PCIe 到 DMA 旁路
启用“PCIe to DMA Bypass”	DMA	PCIe 到 DMA 旁路	

表 4：64 位 BAR

IP 自定义期间的 PCIe BAR 选择	BAR0 (64 位)	BAR2 (64 位)	BAR4 (64 位)
默认值	DMA		
启用“PCIe to AXI Lite Master”	PCIe 到 AXI4-Lite 主接口	DMA	
启用“PCIe to AXI Lite Master”和“PCIe to DMA Bypass”	PCIe 到 AXI4-Lite 主接口	DMA	PCIe 到 DMA 旁路
启用“PCIe to DMA Bypass”	DMA	PCIe 到 DMA 旁路	

可选择 BAR 的不同组合。上表仅列出所有 BAR 的 32 位选择和 64 位选择作为示例。您可基于自己的需求选择不同 BAR 组合。

### 相关信息

[“PCIe BARs”选项卡](#)

## H2C 通道

前述表格演示了 PCIe 到 AXI4-Lite 主接口、PCIe 到 DMA 接口和 PCIe 到 DMA 旁路接口的对应 32 位和 64 位 BAR 选项。针对 32 位和 64 位 BAR 可单独选中每个空间。

H2C 通道数量在 Vivado® 集成设计环境 (IDE) 内进行配置。H2C 通道负责处理从主机到卡的 DMA 传输。它负责根据最大读取请求大小和可用内部资源来拆分读取请求。DMA 通道可保留未完成的请求，其最大数量取决于 `RNUM_RIDS`（即未完成的 H2C 通道请求 ID 参数）。每次拆分（如有）读取请求都会额外耗用一项读取请求。从 DMA 通道向 PCIe RQ 块发出读取后开始，请求即处于未完成状态，直至该通道接收到在用户接口上已按顺序完成写入的确认为止。完成传输后，DMA 通道会发出写回或中断以告知主机传输完成。

H2C 通道还会在其读取接口和写入接口上拆分传输事务。在连接到主机的读取接口上，将根据可用数据 FIFO 空间拆分传输事务以满足配置的最大读取请求大小要求。数据 FIFO 空间在读取请求时进行分配，以确保有足够空间可供完成读取。PCIe RC 块会将完成数据返回到已分配的数据缓冲器位置。为最大程度减小时延，接收到任何完成数据时，H2C 通道都会开始向用户接口发出写入请求。它还会按最大有效载荷大小对写入请求进行分割。在 AXI4-Stream 用户接口上，此拆分操作为透明操作。

启用多个通道时，AXI4 主接口上的传输事务将在所有选中的通道内进行交织。在此情况下会使用简单的循环协议来维护所有通道。传输事务的粒度取决于主机最大有效载荷大小 (MPS)、页面大小和其它主机设置。

## C2H 通道

C2H 通道负责处理从卡到主机的 DMA 传输。C2H 通道的例化数量在 Vivado® IDE 中进行控制。同样，未完成的传输数量通过 `WNUM_RIDS`（即 C2H 通道请求 ID 数量）来配置。在 AXI4-Stream 配置中，先设置 DMA 传输详细信息，然后才会在 AXI4-Stream 接口上接收到数据。这通常是通过接收 DMA 描述符来完成的。在准备好请求 ID 并启用通道后，该通道的 AXI4-Stream 接口即可接收数据并向主机执行 DMA。在 AXI4 MM 接口配置中，向 AXI4 MM 接口发出读取请求时就会分配请求 ID。与 H2C 通道类似，给定请求 ID 保持处于未完成状态，直至完成写入请求为止。对于 C2H 通道，当按 PCIe IP 指示发出写入请求后，写入请求即告完成。

启用多个通道时，AXI4 主接口上的传输事务将在所有选中的通道内进行交织。在此情况下会使用简单的循环协议来维护所有通道。传输事务的粒度取决于主机最大有效载荷大小 (MPS)、页面大小和其它主机设置。

## AXI4-Lite 主接口

此模块用于实现 AXI4-Lite 主接口总线协议。主机可使用此接口来向用户逻辑生成 32 位读取请求和 32 位写入请求。这些读取或写入请求是通过 PCIe 到 AXI4-Lite 主接口 BAR 来接收的。读取完成数据将通过目标桥接器的 PCIe IP CC 总线返回至主机。

## AXI4-Lite 从接口

此模块用于实现 AXI4-Lite 从接口总线协议。用户逻辑只能负责主控该接口上针对 DMA 内部寄存器的 32 位读取或写入操作。您无法通过此接口访问 PCIe 集成块寄存器。此接口不会生成发射到主机的请求。

## 主机到卡旁路主接口

到达 PCIe 到 DMA 旁路 BAR 的主机请求都将被发送到此模块。旁路主端口属于 AXI4 MM 接口，支持读写访问。

## IRQ 模块

IRQ 模块会接收到来自用户逻辑的中断连线，其数量可配置，每个 DMA 通道 1 条中断连线。此模块负责基于 PCIe 生成中断。可在 IP 配置期间指定对 MSI-X、MSI 和传统中断的支持。

**注释：**主机可在 IP 配置期间从指定的受支持中断列表中启用 1 个或多个中断类型。IP 在任意给定时间仅生成 1 种类型的中断，即使启用多个中断类型也是如此。MSI-X 中断优先于 MSI 中断，MSI 中断优先于传统中断。当存在任一已断言有效或暂挂的中断时，主机软件不得切换（启用或禁用）中断类型。

## 传统中断

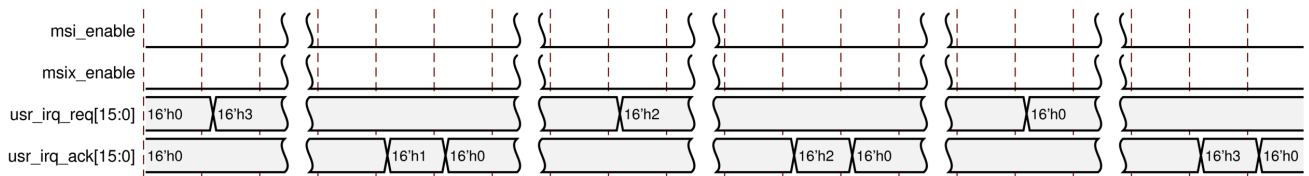
启用传统中断时，usr\_irq\_req 的一个或多个位断言有效会导致 DMA 通过 PCIe 发出传统中断。多个位可同时断言有效，但每个位都必须保持断言有效直至对应的 usr\_irq\_ack 位已断言有效为止。当 usr\_irq\_req 位断言有效后，它必须保持断言有效，直至对应 usr\_irq\_ack 位断言有效，并且主机已处理并清除中断为止。usr\_irq\_ack 断言有效表示请求的中断已发送到 PCIe 块。这将确保主机中断服务例程 (ISR) 在查询 IP 内的中断暂挂寄存器以判定中断源时，此寄存器能保持处于断言有效状态。您必须在用户应用中实现相应的机制才能知晓何时中断例程已处理完成。根据您的应用以及您使用此中断管脚的方式，可通过多种不同方式来执行此检测。在处理中断时，通常需在由主机软件清除、读取或修改的用户应用内实现寄存器（或寄存器阵列）。

当 usr\_irq\_req 位断言无效后，仅当对应 usr\_irq\_ack 位再次断言有效后，它才能重新断言有效。这表示已通过 PCIe 发送对应传统中断的断言无效消息。第二次发生 usr\_irq\_ack 后，xdma0\_usr\_irq\_req 连线可重新断言有效以生成另一次传统中断。

可通过配置寄存器将 xdma0\_usr\_irq\_req 位和 DMA 中断映射到传统中断 INTA、INTB、INTC 和 INTD。下图显示了此传统中断。

**注释：**此图仅显示 xdma0\_usr\_irq\_req 与 usr\_irq\_ack 之间的握手。您的应用可能不会立即清除或处理此中断，在此情况下，您必须使 usr\_irq\_req 保持断言有效直至完成 usr\_irq\_ack 为止。

图 2：传统中断



## MSI 和 MSI-X 中断

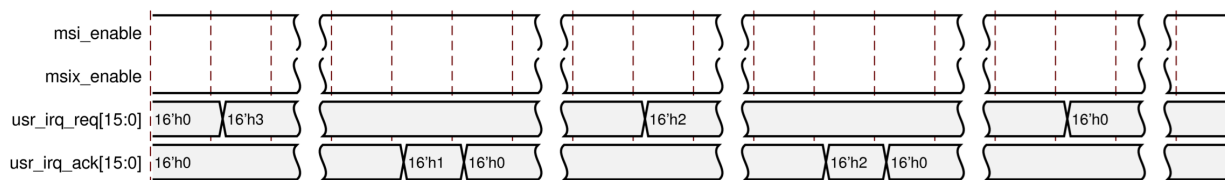
如果启用 MSI 或 MSI-X，那么 usr\_irq\_req 的一个或多个位断言有效会导致生成 MSI 或 MSI-X 中断。如果同时启用 MSI 和 MSI-X 功能，则会生成 MSI-X 中断。

当 usr\_irq\_req 位断言有效后，它必须保持断言有效，直至对应 usr\_irq\_ack 位断言有效，并且主机已处理并清除中断为止。usr\_irq\_ack 断言有效表示请求的中断已发送到 PCIe 块。这将确保主机中断服务例程 (ISR) 在查询 IP 内的中断暂挂寄存器以判定中断源时，此寄存器能保持处于断言有效状态。您必须在用户应用中实现相应的机制才能知晓何时中断例程已处理完成。根据您的应用以及您使用此中断管脚的方式，可通过多种不同方式来执行此检测。在处理中断时，通常需在由主机软件清除、读取或修改的用户应用内实现寄存器（或寄存器阵列）。

配置寄存器可用于将 usr\_irq\_req 和 DMA 中断映射到 MSI 矢量或 MSI-X 矢量。对于 MSI-X 支持，同样有一个矢量表和 PBA 表。下图显示了此 MSI 中断。

**注释：**此图仅显示 usr\_irq\_req 与 usr\_irq\_ack 之间的握手。您的应用可能不会立即清除或处理此中断，在此情况下，您必须使 usr\_irq\_req 保持断言有效直至完成 usr\_irq\_ack 为止。

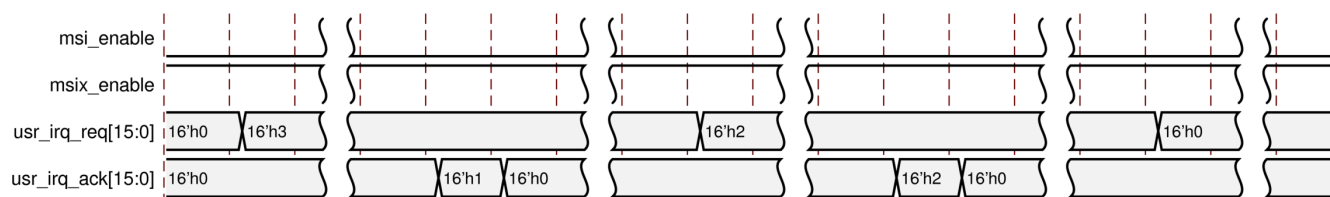
图 3：MSI 中断



下图显示了 MSI-X 中断。

**注释：**此图仅显示 `usr_irq_req` 与 `usr_irq_ack` 之间的握手。您的应用可能不会立即清除或处理此中断，在此情况下，您必须使 `usr_irq_req` 保持断言有效直至完成 `usr_irq_ack` 为止。

图 4：MSI-X 中断



欲知详情，请参阅 [中断处理](#)。

## 配置块

配置模块即包含 PCIe® 解决方案 IP 配置信息和 DMA 控制寄存器的 DMA 寄存器空间，它用于存储与 DMA/Bridge Subsystem for PCIe 相关的 PCIe IP 配置信息。可通过寄存器读取配置模块内的相应寄存器偏移来读取此配置信息。

# DMA 操作

## 快速入门

PCIe® DMA 引擎最根本的作用是在主机存储器与驻留在 FPGA 中的存储器（通常位于插卡上）之间移动数据。将数据从主机存储器移动到 FPGA 存储器上的过程称为主机到卡 (Host to Card, H2C) 传输或系统到卡 (System to Card, S2C) 传输。相反，将数据从 FPGA 存储器移动到主机存储器的过程则被称为卡到主机 (Card to Host, C2H) 传输或卡到系统 (Card to System, C2S) 传输。

这些术语有助于澄清数据流动方向，相反，若使用读取和写入这两个术语则很容易引发混淆。PCIe DMA 引擎仅用于在不同 PCIe 地址位置之间移动数据。

在典型操作中，主机中的应用必须在 FPGA 与主机存储器之间移动数据。为了完成此传输，主机会在系统存储器中设置缓冲空间，并创建描述符以供 DMA 引擎用于移动数据。

描述符的内容将取决于多种因素，包括针对 DMA 引擎所选的用户接口等。如果选择 AXI4-Stream 接口，那么 C2H 传输不使用源地址字段，H2C 字段则不使用目标地址。这是因为 AXI4-Stream 接口为 FIFO 型接口，不使用地址。



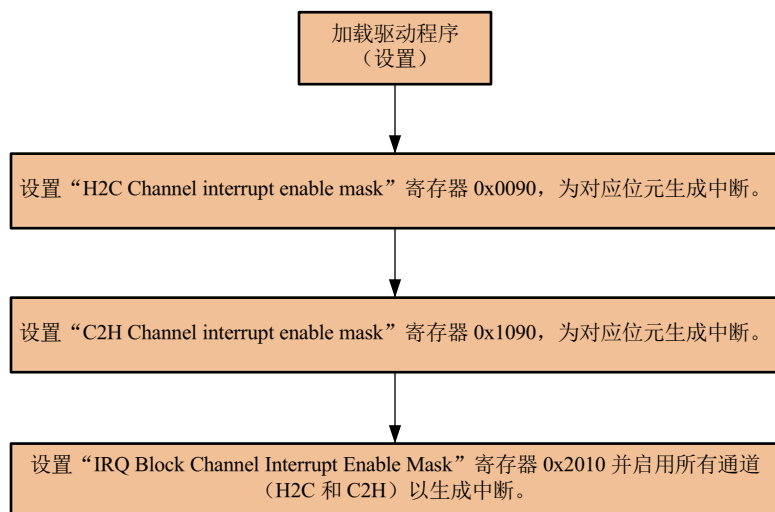
如果选择 AXI Memory Mapped 接口，那么 C2H 传输使用源地址作为 AXI 地址，目标地址即为 PCIe 地址。对于 H2C 传输，源地址为 PCIe 地址，目标地址则是 AXI 地址。

以下流程图演示了在对 AXI Memory Mapped 接口进行 IP 配置期间选中数据接口的情况下，H2C 和 C2H 传输的典型流程。

## H2C 传输和 C2H 传输的初始设置

下图显示了 H2C 传输和 C2H 传输的初始设置。

图 5：设置

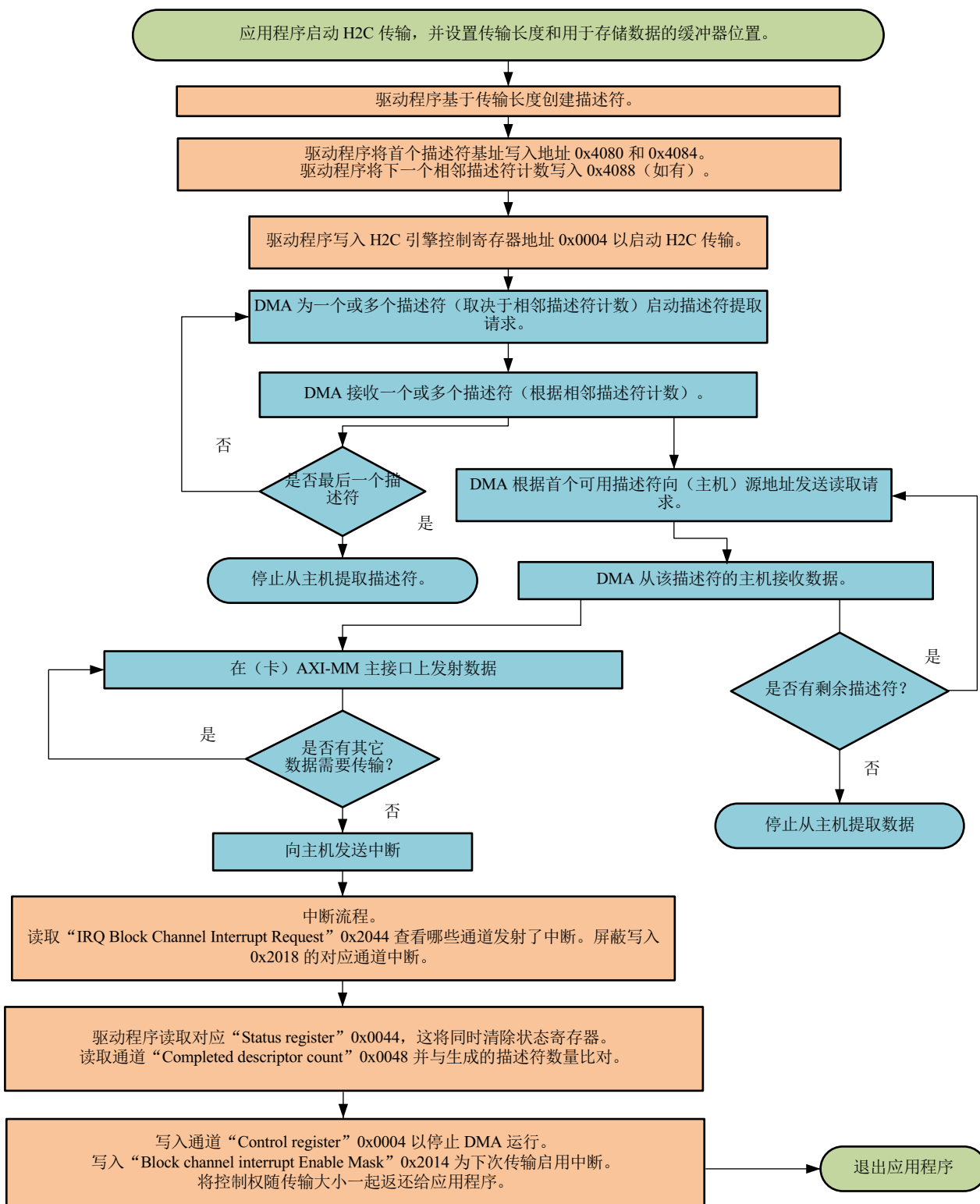


X19438-051621

## 适用于 H2C 的 AXI-MM 传输

下图所示基本流程图解释了 H2C 的数据传输。流程图颜色编码如下：绿色表示应用程序，橙色表示驱动程序，蓝色表示硬件。

图 6：DMA H2C 传输摘要

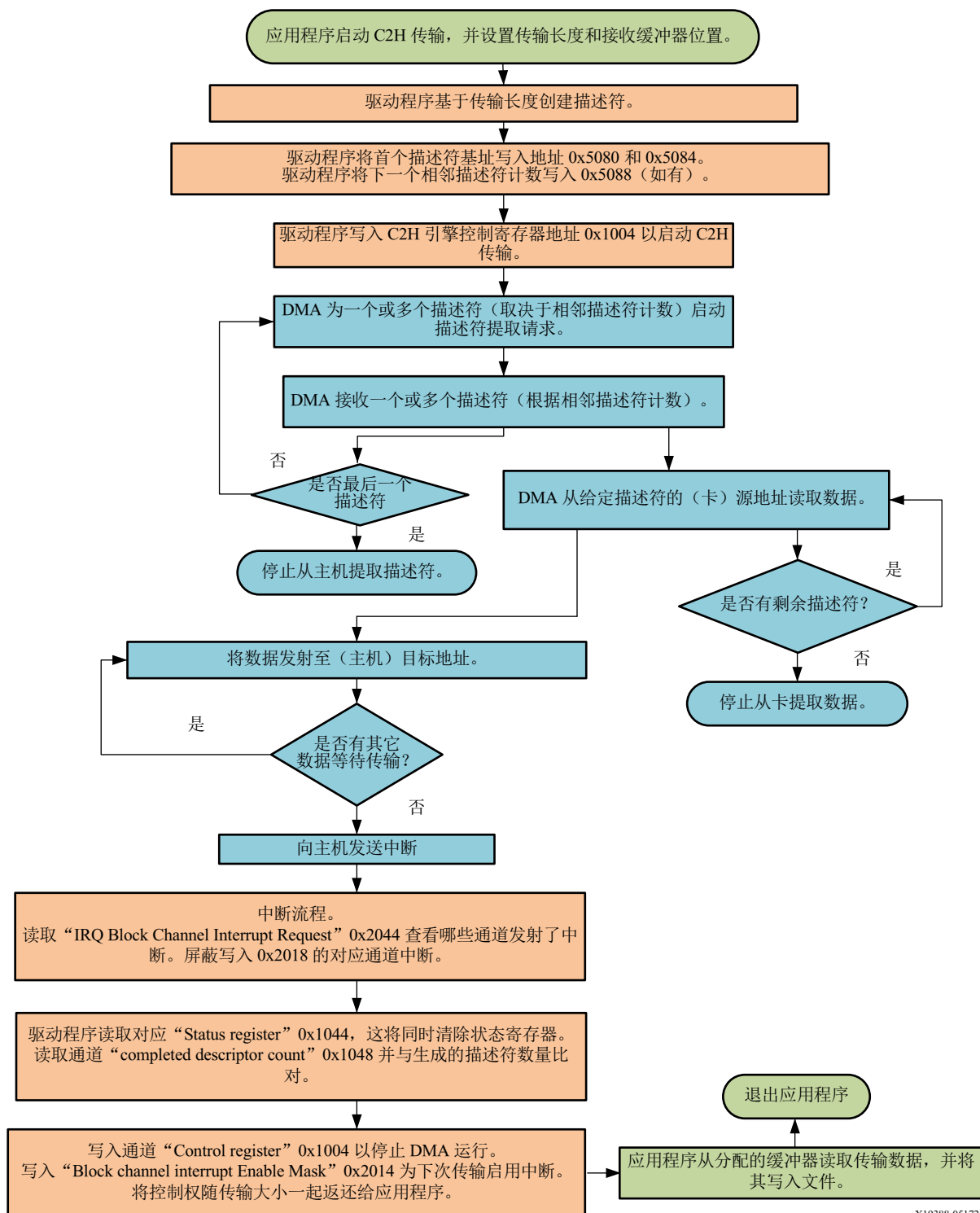


X19389-051621

## 适用于 C2H 的 AXI-MM 传输

下图所示基本流程图解释了 C2H 的数据传输。流程图颜色编码如下：绿色表示应用程序，橙色表示驱动程序，蓝色表示硬件。

图 7：DMA C2H 传输摘要



X19388-051721

## 描述符

DMA/Bridge Subsystem for PCI Express® 使用已链接的描述符列表来指定 DMA 传输的源、目标和长度。描述符列表是由驱动程序创建的，存储在主机存储器内。DMA 通道由含数个控制寄存器的驱动程序进行初始化，以开始提取描述符列表并执行 DMA 操作。

描述符用于描述 DMA/Bridge Subsystem for PCIe 应执行的存储器传输。每个通道都有其自己的描述符列表。每个通道的描述符列表的起始地址均在硬件寄存器中由驱动程序进行初始化。启用通道后，描述符通道就会开始从初始地址提取描述符。随后，它会从已提取的最后一个描述符的 `Nxt_adr[63:0]` 字段中执行提取。描述符必须对齐到 32 个字节的边界处。

相邻描述符的初始块的大小是使用 `Dsc_Adj` 寄存器来指定的。初始提取后，描述符通道使用最后一个提取的描述符的 `Nxt_adj` 字段来判定下一个描述符地址的描述符数量。相邻描述符块不得跨 4K 地址边界。描述符通道会在单一请求内提取尽可能多的描述符，可提取的数量受 MRRS、相邻描述符数量以及通道的描述符缓冲器内的可用空间所限。

**注释：**由于 MRRS 在大部分主机系统中为 512 个字节或 1024 个字节，因此在单一请求中不允许相邻描述符数量超过 32 个。但如果需要，设计在相邻描述符构成的单一块内允许的描述符最大数量为 64 个。

描述符列表中的每个描述符都必须准确描述紧随其后的描述符或描述符块。在相邻描述符构成的块中，`Nxt_adj` 值从第一个描述符开始递减直至倒数第二个描述符（值为 0）。同样，块中的每个描述符指向块中的下一个描述符，最后一个描述符除外，它可能指向新的块或者可能使列表终止。

描述符列表终止以停止 (Stop) 控制位来表示。观测到含 Stop 控制位的描述符后，针对该列表将不再发出描述符提取请求。Stop 控制位只能在块的最后一个描述符上置位。

使用 AXI4 Memory Mapped 接口时，不会对卡的 DMA 地址执行转换。如果主机不知道卡的地址映射，则必须在用户逻辑中汇编描述符，并使用描述符旁路接口将其提交至 DMA。

表 5：描述符格式

偏移	字段			
0x0	Magic[15:0]	Rsv[1:0]	Nxt_adj[5:0]	Control[7:0]
0x04	4'h0, Len[27:0]			
0x08	Src_adr[31:0]			
0x0C	Src_adr[63:32]			
0x10	Dst_adr[31:0]			
0x14	Dst_adr[63:32]			
0x18	Nxt_adr[31:0]			
0x1C	Nxt_adr[63:32]			

表 6：描述符字段

偏移	字段	位索引	子字段	描述
0x0	Magic	15:0		16'had4b。此代码用于验证驱动程序生成的描述符是否有效。
0x0		1:0		保留，置位为 0

表 6：描述符字段 (续)

偏移	字段	位索引	子字段	描述
0x0	Nxt_adj	5:0		位于下一个描述符地址 字段所在处的描述符之 后的额外相邻描述符的 数量。 相邻描述符块不得跨 4k 边界。
0x0	Control	5、6、7		保留
0x0		4	EOP	针对 Stream 接口的包 结束。
0x0		2 和 3		保留
0x0		1	Completed	设为 1 即可在引擎完成 此描述符后中断。这需 要 在 H2C/C2H 通道控制寄 存器中设置 IE_DESCRIPTOR_COMP LETED 控制标记。
0x0		0	Stop	设为 1 即可为此描述符 列表停止提取描述符。 Stop 位只能在相邻描述 符块的最后一个描述符 上置位。
0x04	Length	31:28		保留，置位为 0
0x04		27:0		数据长度（以字节为单 位）。
0x0C-0x8	Src_adr	63:0		H2C 和存储器映射传输 的源地址。 C2H 传输的元数据写回 地址。
0x14-0x10	Dst_adr	63:0		C2H 和存储器映射传输 的目标地址。不可用于 H2C 数据流传输。
0x1C-0x18	Nxt_adr	63:0		列表中下一个描述符的 地址。

DMA 包含深度为  $\text{Bit\_width} * 512$  的 FIFO，用于保存描述符引擎中的所有描述符。此描述符 FIFO 与所有选定的通道共享。

- 对于含 2H2C 和 2C2H 的 Gen3x8 设计，AXI 位宽为 256 位。FIFO 深度为  $256 \text{ bit} * 512 = 32 \text{ B} * 512 = 16 \text{ KB}$ （512 个描述符）。此 FIFO 供 4 个 DMA 引擎共享。

## 描述符旁路

根据通道，可通过 Vivado® IDE 参数来绕过描述符提取引擎。启用描述符旁路的通道可从其相应的 `c2h_dsc_byp` 或 `h2c_dsc_byp` 总线接受描述符。在通道接受描述符前，控制寄存器的“运行 (Run)”位必须置位。绕过描述符时，不使用 `NextDescriptorAddress`、`NextAdjacentCount` 和 `Magic` 描述符字段。控制寄存器位中的 `ie_descriptor_stopped` 位不会阻止用户逻辑写入其它描述符。写入通道的所有描述符都会接受处理，当通道缓冲器已满时，将限制写入新描述符。

## 轮询模式

每个引擎都能将已完成的描述符计数写回主机存储器。这样驱动程序即可轮询主机存储器以判定何时 DMA 完成，而无需等待中断。

对于给定 DMA 引擎，当 DMA 完成描述符传输并且 `ie_descriptor_completed` 和 `Pollmode_wb_enable` 均已置位后，就会发生已完成的描述符计数写回操作。报告的已完成描述符计数是从 DMA 启动开始，已完成的描述符总数（而不只是含“Completed”标记的描述符置位）。写回地址由 `Pollmode_hi_wb_addr` 和 `Pollmode_lo_wb_addr` 寄存器来定义。

表 7：已完成的描述符计数写回格式

偏移	字段		
0x0	Sts_err	7'h0	Compl_descriptor_count[23:0]

表 8：已完成的描述符计数写回字段

字段	描述
Sts_err	通道状态寄存器内任意错误状态位的按位 OR。
Compl_descriptor_count[23:0]	Complete Descriptor Count 寄存器的下 24 位。

## DMA H2C 数据流传输

对于主机到卡传输，将从位于源地址的主机读取数据，但不使用描述符中的目标地址。数据包可跨多个描述符。包终止以 EOP 控制位来表示。具有 EOP 位的描述符会在数据最后一拍的 AXI4-Stream 用户接口上断言 `tlast` 有效。

交付至 AXI4-Stream 接口的数据将按描述符逐个进行打包，`tkeep` 全部为 1，但如果描述符长度并非数据路径宽度的整数倍，则其数据传输的最后一个周期内，`tkeep` 并非全部为 1。DMA 不会跨多个描述符进行数据打包。

## DMA C2H 数据流传输

对于卡到主机传输，数据接收自 AXI4-Stream 接口并写入目标地址。数据包可跨多个描述符。C2H 通道启用时即可接受数据，并具有有效的描述符。接收到数据后，就会按顺序填充描述符。当描述符完成填充或者由于接口上出现包结束而导致描述符关闭后，C2H 通道会将信息写回含预定义的 WB Magic 值 `16'h52b4` (表 10：C2H 数据流传输写回字段) 的主机上的写回地址，并对 EOP 和长度 (Length) 进行相应的更新。对于 C2H AXI4-Stream 接口上的有效数据周期，与给定数据包关联的所有数据都必须连续。

**注释：**C2H 通道写回信息不同于轮询模式更新。C2H 通道写回信息可提供特定描述符的驱动程序当前长度状态。这不同于 [轮询模式](#) 中所述的 `Pollmode_*`。

对于任一数据包的所有传输（最后一次数据传输除外）而言，`tkeep` 位必须全部为 1。在执行数据包的最后一次传输时，当 `tlast` 断言有效时，可指定 `tkeep` 不全为 1，以指定该数据周期并非完整的数据路径宽度。断言有效的 `tkeep` 位需打包到 `lsb` 中以表示连续数据。

C2H 数据流传输描述符的长度（目标缓冲器的大小）必须始终为 64 个字节的倍数。

表 9：C2H 数据流传输写回格式

偏移	字段		
0x0	WB Magic[15:0]	Reserved [14:0]	Status[0]

表 9：C2H 数据流传输写回格式 (续)

偏移	字段
0x04	Length[31:0]

表 10：C2H 数据流传输写回字段

字段	位索引	子字段	描述
Status	0	EOP	包结束
Reserved	14:0		保留
WB Magic	15:0		16'h52b4。此代码用于验证 C2H 写回是否有效。
Length	31:0		数据长度（以字节为单位）。

## 地址对齐

表 11：地址对齐

接口类型	数据路径宽度	地址限制
AXI4 MM	64、128、256 或 512	无
AXI4-Stream	64、128、256 或 512	无
AXI4 MM 固定地址 <sup>1</sup>	64	Source_addr[2:0] == Destination_addr[2:0] == 3'h0
AXI4 MM 固定地址 <sup>1</sup>	128	Source_addr[3:0] == Destination_addr[3:0] == 4'h0
AXI4 MM 固定地址 <sup>1</sup>	256	Source_addr[4:0] == Destination_addr[4:0] == 5'h0
AXI4 MM 固定地址 <sup>1</sup>	512	Source_addr[5:0] == Destination_addr[5:0] == 6'h0

注释：

- 对于固定地址模式，必须在控制寄存器中将位 [25] 置位。

### 相关信息

[H2C Channel Control \(0x04\)](#)

[C2H Channel Control \(0x04\)](#)

## 长度粒度

表 12：长度粒度

接口类型	数据路径宽度	长度粒度限制
AXI4 MM	64、128、256 或 512	无
AXI4-Stream	64、128、256 或 512	无 <sup>1</sup>
AXI4 MM 固定地址	64	Length[2:0] == 3'h0
AXI4 MM 固定地址	128	Length[3:0] == 4'h0
AXI4 MM 固定地址	256	Length[4:0] == 5'h0



表 12：长度粒度 (续)

接口类型	数据路径宽度	长度粒度限制
AXI4 MM 固定地址	512	Length[5:0] == 6'h0

**注释：**

- 每个 C2H 描述符大小都必须为 64 个字节的倍数。但对于实际 C2H 传输中的字节总数并无限制。

## 奇偶校验

奇偶校验检查可采用以下 2 种方法之一。在核自定义期间，在 Vivado® IDE 的“PCIe DMA”选项卡下设置“Parity Checking”选项：

启用“Check Parity”后，DMA/Bridge Subsystem for PCIe 会在读取来自 PCIe 的数据时执行奇偶校验检查，并在将数据写入 PCIe 时生成奇偶校验。

启用“Propagate Parity”时，DMA/Bridge Subsystem for PCIe 会将奇偶校验传输至用户 AXI 接口。您负责在 AXI 接口中检查和生成奇偶校验。在数据有效信号断言有效的每个时钟周期内，奇偶校验都有效，并且奇偶校验位仅对有效的数据字节有效。针对每个字节都会计算奇偶校验；奇偶校验总位数为 DATA\_WIDTH/8。

- 在 AXI4-Stream (AXI\_ST) 模式下，在 \*\_tuser 端口上发送和接收奇偶校验信息。
- 在 AXI4 Memory Mapped (AXI-MM) 模式下，在 \*\_ruser 和 \*\_wuser 端口上发送和接收奇偶校验信息。

针对奇偶校验检查使用奇校验。默认情况下，不启用奇偶校验检查。

**相关信息**

[“PCIe DMA”选项卡](#)

## 端口描述



**重要提示！** 本文档仅涵盖 DMA 模式端口描述。对于 AXI Bridge 模式，请参阅《AXI Bridge for PCI Express Gen3 Subsystem 产品指南》(PG194)。

DMA/Bridge Subsystem for PCI Express® 直接连接至 Integrated Block for PCIe。根据 IP 配置，连接到 PCIe 集成块 IP 的数据路径接口位宽为 64、128、256 或 512 位，且运行速率上限为 250 MHz。数据路径宽度适用于除 AXI4-Lite 接口外的所有数据接口。AXI4-Lite 接口位宽固定为 32 位。

下列各表描述了与该子系统关联的端口。

## XDMA 全局端口

表 13：顶层接口信号

信号名称	方向	描述
sys_clk	输入	7 系列 Gen2 和 Virtex-7 Gen3: PCIe 参考时钟。应从参考时钟 IBUFDS_GTE2 的 O 端口驱动。 UltraScale: DRP 时钟和内部系统时钟（如果 PCIe 参考时钟为 250 MHz，则频率为 sys_clk_gt 的一半，否则与 sys_clk_gt 频率相同）。应从参考时钟 IBUFDS_GTE3 的 ODIV2 端口驱动。
sys_clk_gt	输入	仅限 UltraScale: PCIe 参考时钟。应从参考时钟 IBUFDS_GTE3 的 O 端口驱动。请参阅《UltraScale 器件 Gen3 Integrated Block for PCI Express LogiCORE IP 产品指南》(PG156) 或《UltraScale+ Integrated Block for PCI Express LogiCORE IP 产品指南》(PG213)。
sys_rst_n	输入	从 PCIe 边缘连接器复位信号进行复位
axi_aclk	输出	PCIe 为 m_axi* 接口和 s_axi* 接口衍生的时钟输出。axi_aclk 为来自 TXOUTCLK 管脚（源于 GT 块）的衍生时钟；当 axi_aresetn 断言有效时，该衍生时钟不应持续运行。
axi_aresetn	输出	AXI 复位信号，与 axi_aclk 输出上提供的时钟同步。此复位应驱动所有对应的 AXI Interconnect aresetn 信号。
dma_bridge_resetn	输入	可选管脚，仅当 SOFT_RESET_EN 参数设置为 TRUE 时才可用。发生链路中断、功能级复位 (Function Level Reset)、Dynamic Function eXchange 或用户定义的其它错误状况时，此管脚应作为用户驱动的复位来使用。初始链路正常运行期间，无需切换此管脚。 使用时，所有 PCIe 流量都必须处于静止状态。此信号必须断言有效，并且保持时间必须超过“完成超时 (Completion Timeout)”值（通常为 50 ms）。 <ul style="list-style-type: none"> <li>0: 复位所有内部 Bridge 引擎和寄存器，并将 axi_aresetn 信号断言有效，同时保持 PCIe 链路正常运行。</li> <li>1: 正常操作。</li> </ul> 请参阅“时钟设置和复位”以获取有关使用此信号的更多指示信息。
user_lnk_up	输出	当输出处于高电平有效时，表示 PCI Express 核已正常链接至主机器件。
msi_enable	输出	表示何时启用 MSI。
msi_vector_width[2:0]	输出	表示 MSI 字段的大小（分配到器件的 MSI 矢量数）。
msix_enable	输出	表示何时启用 MSI-X。

### 相关信息

[时钟设置和复位](#)

## PCIe 接口信号

表 14：PCIe 接口信号

信号名称	方向	描述
pci_exp_rxp[PL_LINK_CAP_MAX_LINK_WIDTH-1:0]	输入	PCIe RX 串行接口
pci_exp_rxn[PL_LINK_CAP_MAX_LINK_WIDTH-1:0]	输入	PCIe RX 串行接口
pci_exp_txp[PL_LINK_CAP_MAX_LINK_WIDTH-1:0]	输出	PCIe TX 串行接口
pci_exp_txn[PL_LINK_CAP_MAX_LINK_WIDTH-1:0]	输出	PCIe TX 串行接口

## H2C 通道 0-3 AXI4-Stream 接口信号

表 15: H2C 通道 0-3 AXI4-Stream 接口信号

信号名称 <sup>1</sup>	方向	描述
m_axis_h2c_tready_x	输入	用户逻辑断言此信号有效表示它已准备好接受数据。当在同一周期内 m_axis_h2c_tready 和 m_axis_h2c_tvalid 均断言有效时，就会通过该接口传输数据。当有效信号为高电平时，如果用户逻辑断言此信号无效，那么 DMA 会使有效信号保持处于断言有效状态，直至就绪信号断言有效为止。
m_axis_h2c_tlast_x	输出	DMA 在 DMA 包的最后一个节拍内断言此信号有效以指示包结束。
m_axis_h2c_tdata_x [DATA_WIDTH-1:0]	输出	将数据从 DMA 发射到用户逻辑。
m_axis_h2c_tvalid_x	输出	当 DMA 在 m_axis_h2c_tdata 上驱动有效数据时，始终断言此信号有效。
m_axis_h2c_tuser_x [DATA_WIDTH/8-1:0]	输出	奇偶校验位。此端口仅在“传输奇偶校验 (Propagate Parity)”模式下才启用。

**注释：**

1. 信号名称中的 \_x 会根据通道编号（0、1、2 和 3）而变。例如，对于通道 0，使用 m\_axis\_h2c\_tready\_0 端口，对于通道 1，则使用 m\_axis\_h2c\_tready\_1 端口。

## C2H 通道 0-3 AXI4-Stream 接口信号

表 16: C2H 通道 0-3 AXI4-Stream 接口信号

信号名称 <sup>1</sup>	方向	描述
s_axis_c2h_tready_x	输出	此信号断言有效表示 DMA 已准备好接受数据。当在同一周期内 s_axis_c2h_tready 和 s_axis_c2h_tvalid 均断言有效时，就会通过该接口传输数据。当有效信号为高电平时，如果 DMA 断言此信号无效，那么用户逻辑必须使有效信号保持处于断言有效状态，直至就绪信号断言有效为止。
s_axis_c2h_tlast_x	输入	用户逻辑断言此信号有效，以指示 DMA 包结束。
s_axis_c2h_tdata_x [DATA_WIDTH-1:0]	输入	将数据从用户逻辑发射到 DMA。
s_axis_c2h_tvalid_x	输入	当用户逻辑在 s_axis_c2h_tdata 上驱动有效数据时，始终断言此信号有效。
m_axis_c2h_tuser_x [DATA_WIDTH/8-1:0]	输入	奇偶校验位。此端口仅在“传输奇偶校验 (Propagate Parity)”模式下才启用。

**注释：**

1. 信号名称中的 \_x 会根据通道编号（0、1、2 和 3）而变。例如，对于通道 0，使用 m\_axis\_c2h\_tready\_0 端口，对于通道 1，则使用 m\_axis\_c2h\_tready\_1 端口。

## AXI4 Memory Mapped 读取地址接口信号

表 17: AXI4 Memory Mapped 读取地址接口信号

信号名称	方向	描述
m_axi_araddr [AXI_ADR_WIDTH-1:0]	输出	此信号为存储器映射读取地址（从 DMA 到用户逻辑）。
m_axi_arid [ID_WIDTH-1:0]	输出	标准 AXI4 描述，欲知详情，请参阅 <a href="#">AXI4 协议规范</a> 。
m_axi_arlen[7:0]	输出	主读取突发长度。
m_axi_arsize[2:0]	输出	主读取突发大小。
m_axi_arprot[2:0]	输出	3'h0
m_axi_arvalid	输出	此信号断言有效即表示存在发射到 m_axi_araddr 上的地址的有效读取请求。
m_axi_arready	输入	主读取地址就绪。
m_axi_arlock	输出	1'b0
m_axi_arcache[3:0]	输出	4'h0
m_axi_arburst	输出	主读取突发类型。

## AXI4 Memory Mapped 读取接口信号

表 18: AXI4 Memory Mapped 读取接口信号

信号名称	方向	描述
m_axi_rdata [DATA_WIDTH-1:0]	输入	主读取数据。
m_axi_rid [ID_WIDTH-1:0]	输入	主读取 ID。
m_axi_rresp[1:0]	输入	主读取响应。
m_axi_rlast	输入	主读取结束。
m_axi_rvalid	输入	主读取有效。
m_axi_rready	输出	主读取就绪。
m_axi_ruser [DATA_WIDTH/8-1:0]	输入	读取接口的奇偶校验端口。此端口仅在“传输奇偶校验 (Propagate Parity)”模式下才启用。

## AXI4 Memory Mapped 写入地址接口信号

表 19: AXI4 Memory Mapped 写入地址接口信号

信号名称	方向	描述
m_axi_awaddr [AXI_ADR_WIDTH-1:0]	输出	此信号为存储器映射写入地址（从 DMA 到用户逻辑）。
m_axi_awid [ID_WIDTH-1:0]	输出	主写入地址 ID。
m_axi_awlen[7:0]	输出	主写入地址长度。
m_axi_awsz[2:0]	输出	主写入地址大小。
m_axi_awburst[1:0]	输出	主写入地址突发类型。

表 19: AXI4 Memory Mapped 写入地址接口信号 (续)

信号名称	方向	描述
m_axi_awprot[2:0]	输出	3'h0
m_axi_awvalid	输出	此信号断言有效即表示存在发射到 m_axi_araddr 上的地址的有效写入请求。
m_axi_awready	输入	主写入地址就绪。
m_axi_awlock	输出	1'b0
m_axi_awcache[3:0]	输出	4'h0

## AXI4 Memory Mapped 写入接口信号

表 20: AXI4 Memory Mapped 写入接口信号

信号名称	方向	描述
m_axi_wdata [DATA_WIDTH-1:0]	输出	主写入数据。
m_axi_wlast	输出	主写入结束。
m_axi_wstrb	输出	主写入选通。
m_axi_wvalid	输出	主写入有效。
m_axi_wready	输入	主写入就绪。
m_axi_wuser [DATA_WIDTH/8-1:0]	输出	读取接口的奇偶校验端口。此端口仅在“传输奇偶校验 (Propagate Parity)”模式下才启用。

## AXI4 Memory Mapped 写入响应接口信号

表 21: AXI4 Memory Mapped 写入响应接口信号

信号名称	方向	描述
m_axi_bvalid	输入	主写入响应有效。
m_axi_bresp[1:0]	输入	主写入响应。
m_axi_bid [ID_WIDTH-1:0]	输入	主响应 ID。
m_axi_bready	输出	主响应就绪。

## AXI4 Memory Mapped 主旁路读取地址接口信号

表 22: AXI4 Memory Mapped 主旁路读取地址接口信号

信号名称	方向	描述
m_axib_araddr [AXI_ADR_WIDTH-1:0]	输出	此信号为存储器映射读取地址（从主机到用户逻辑）。
m_axib_arid [ID_WIDTH-1:0]	输出	主读取地址 ID。
m_axib_arlen[7:0]	输出	主读取地址长度。

表 22: AXI4 Memory Mapped 主旁路读取地址接口信号 (续)

信号名称	方向	描述
m_axib_arsize[2:0]	输出	主读取地址大小。
m_axib_arprot[2:0]	输出	3'h0
m_axib_arvalid	输出	此信号断言有效即表示存在发射到 m_axib_araddr 上的地址的有效读取请求。
m_axib_arready	输入	主读取地址就绪。
m_axib_arlock	输出	1'b0
m_axib_arsize[3:0]	输出	4'h0
m_axib_arburst	输出	主读取地址突发类型。

## AXI4 Memory Mapped 主旁路读取接口信号

表 23: AXI4 Memory Mapped 主旁路读取接口信号

信号名称	方向	描述
m_axib_rdata [DATA_WIDTH-1:0]	输入	主读取数据。
m_axib_rid [ID_WIDTH-1:0]	输入	主读取 ID。
m_axib_rresp[1:0]	输入	主读取响应。
m_axib_rlast	输入	主读取结束。
m_axib_rvalid	输入	主读取有效。
m_axib_rready	输出	主读取就绪。
m_axib_ruser [DATA_WIDTH/8-1:0]	输入	读取接口的奇偶校验端口。此端口仅在“传输奇偶校验 (Propagate Parity)”模式下才启用。

## AXI4 Memory Mapped 主旁路写入地址接口信号

表 24: AXI4 Memory Mapped 主旁路写入地址接口信号

信号名称	方向	描述
m_axib_awaddr [AXI_ADR_WIDTH-1:0]	输出	此信号为存储器映射写入地址（从主机到用户逻辑）。
m_axib_awid [ID_WIDTH-1:0]	输出	主写入地址 ID。
m_axib_awlen[7:0]	输出	主写入地址长度。
m_axib_awsz[2:0]	输出	主写入地址大小。
m_axib_awburst[1:0]	输出	主写入地址突发类型。
m_axib_awprot[2:0]	输出	3'h0
m_axib_awvalid	输出	此信号断言有效即表示存在发射到 m_axib_araddr 上的地址的有效写入请求。
m_axib_awready	输入	主写入地址就绪。
m_axib_awlock	输出	1'b0

表 24: AXI4 Memory Mapped 主旁路写入地址接口信号 (续)

信号名称	方向	描述
m_axib_awcache[3:0]	输出	4'h0

## AXI4 Memory Mapped 主旁路写入接口信号

表 25: AXI4 Memory Mapped 主旁路写入接口信号

信号名称	方向	描述
m_axib_wdata [DATA_WIDTH-1:0]	输出	主写入数据。
m_axib_wlast	输出	主写入结束。
m_axib_wstrb	输出	主写入选通。
m_axib_wvalid	输出	主写入有效。
m_axib_wready	输入	主写入就绪。
m_axib_wuser [DATA_WIDTH/8-1:0]	输出	读取接口的奇偶校验端口。此端口仅在“传输奇偶校验 (Propagate Parity)”模式下才启用。

## AXI4 Memory Mapped 主旁路写入响应接口信号

表 26: AXI4 Memory Mapped 主旁路写入响应接口信号

信号名称	方向	描述
m_axib_bvalid	输入	主写入响应有效。
m_axib_bresp[1:0]	输入	主写入响应。
m_axib_bid [ID_WIDTH-1:0]	输入	主写入响应 ID。
m_axib_bready	输出	主响应就绪。

## 配置 AXI4-Lite Memory Mapped 写入主接口信号

表 27: 配置 AXI4-Lite Memory Mapped 写入主接口信号

信号名称	方向	描述
m_axil_awaddr[31:0]	输出	此信号为存储器映射写入地址（从主机到用户逻辑）。
m_axil_awprot[2:0]	输出	3'h0
m_axil_awvalid	输出	此信号断言有效即表示存在发射到 m_axil_awaddr 上的地址的有效写入请求。
m_axil_awready	输入	主写入地址就绪。
m_axil_wdata[31:0]	输出	主写入数据。
m_axil_wstrb	输出	主写入选通。
m_axil_wvalid	输出	主写入有效。
m_axil_wready	输入	主写入就绪。

表 27：配置 AXI4-Lite Memory Mapped 写入主接口信号 (续)

信号名称	方向	描述
m_axil_bvalid	输入	主响应有效。
m_axil_bready	输出	主响应就绪。

## 配置 AXI4-Lite Memory Mapped 读取主接口信号

表 28：配置 AXI4-Lite Memory Mapped 读取主接口信号

信号名称	方向	描述
m_axil_araddr[31:0]	输出	此信号为存储器映射读取地址（从主机到用户逻辑）。
m_axil_arprot[2:0]	输出	3'h0
m_axil_arvalid	输出	此信号断言有效即表示存在发射到 m_axil_araddr 上的地址的有效读取请求。
m_axil_arready	输入	主读取地址就绪。
m_axil_rdata[31:0]	输入	主读取数据。
m_axil_rresp	输入	主读取响应。
m_axil_rvalid	输入	主读取有效。
m_axil_rready	输出	主读取就绪。

## 配置 AXI4-Lite Memory Mapped 写入从接口信号

表 29：配置 AXI4-Lite Memory Mapped 写入从接口信号

信号名称	方向	描述
s_axil_awaddr[31:0]	输入	此信号为存储器映射写入地址（从用户逻辑到 DMA）。
s_axil_awvalid	输入	此信号断言有效即表示存在发射到 s_axil_awaddr 上的地址的有效写入请求。
s_axil_awprot[2:0]	输入	未使用
s_axil_awready	输出	从写入地址就绪。
s_axil_wdata[31:0]	输入	从写入数据。
s_axil_wstrb	输入	从写入选通。
s_axil_wvalid	输入	从写入有效。
s_axil_wready	输出	从写入就绪。
s_axil_bvalid	输出	从写入响应有效。
s_axil_bready	输入	保存响应就绪。



## 配置 AXI4-Lite Memory Mapped 读取从接口信号

表 30：配置 AXI4-Lite Memory Mapped 读取从接口信号

信号名称	方向	描述
s_axil_araddr[31:0]	输入	此信号为存储器映射读取地址（从用户逻辑到 DMA）。
s_axil_arprot[2:0]	输入	未使用
s_axil_arvalid	输入	此信号断言有效即表示存在发射到 s_axil_araddr 上的地址的有效读取请求。
s_axil_arready	输出	从读取地址就绪。
s_axil_rdata[31:0]	输出	从读取地址。
s_axil_rresp	输出	从读取响应。
s_axil_rvalid	输出	从读取有效。
s_axil_rready	输入	从读取就绪。

## 中断接口

表 31：中断接口

信号名称	方向	描述
usr_irq_req[NUM_USR_IRQ-1:0]	输入	此信号断言有效即生成中断。保持断言有效直至中断处理完成为止。
usr_irq_ack[NUM_USR_IRQ-1:0]	输出	表示在 PCIe 上已发送中断。针对传统中断会生成 2 次确认。针对 MSI 中断会生成 1 次确认。

usr\_irq\_req 总线中的每个位元都对应于 usr\_irq\_ack 中的相同位元。例如，usr\_irq\_ack[0] 表示对应 usr\_irq\_req[0] 的确认。

## 通道 0-3 状态端口

表 32：通道 0-3 状态端口

信号名称	方向	描述
h2c_sts [7:0]	输出	每个通道的状态位。位： 6：控制寄存器“运行 (Run)”位 5：IRQ 事件暂挂 4：数据包完成事件 (AXI4-Stream) 3：描述符完成事件。针对完成的每个描述符脉冲 1 个周期，与 Descriptor.Completed 字段无关 2：状态寄存器 Descriptor_stop 位 1：状态寄存器 Descriptor_completed 位 0：状态寄存器“繁忙 (busy)”位

表 32：通道 0-3 状态端口 (续)

信号名称	方向	描述
c2h_sts [7:0]	输出	每个通道的状态位。位： 6：控制寄存器“运行 (Run)”位 5：IRQ 事件暂挂 4：数据包完成事件 (AXI4-Stream) 3：描述符完成事件。针对完成的每个描述符脉冲 1 个周期，与 Descriptor.Completed 字段无关 2：状态寄存器 Descriptor_stop 位 1：状态寄存器 Descriptor_completed 位 0：状态寄存器“繁忙 (busy)”位

## 配置扩展接口端口描述

在实现外部实现的配置寄存器时，“配置扩展 (Configuration Extend)”接口允许核随用户应用一起传输配置信息。下表定义了核的配置扩展接口中的端口。

**注释：**此接口对于 7 系列 Gen2 IP 不可用。

表 33：配置扩展接口端口描述

端口	方向	宽度	描述
cfg_ext_read_received	输出	1	已接收配置扩展读取 核接收到来自链路的配置读取请求时，会断言此输出有效。当启用户实现的传统空间或扩展配置空间时，接收到配置读取会导致此信号断言有效并保持 1 个周期，同时生成有效的 <code>cfg_ext_register_number</code> 和 <code>cfg_ext_function_number</code> 。 当启用户实现的传统空间和/或扩展配置空间时，如果 <code>cfg_ext_register_number</code> 下降至指定范围以下，则此信号会断言有效，并且用户逻辑必须提供 <code>cfg_ext_read_data</code> 和 <code>cfg_ext_read_data_valid</code> 。 传统空间： 0xB0-0xBF 扩展配置空间： 0xE80 - 0xFFF (仅限 UltraScale+ HBM PCIe4C 核) 0x480 - 0x4FF
cfg_ext_write_received	输出	1	已接收配置扩展写入 该核接收到来自链路的配置写入请求时，会在此输出上生成 1 个周期的脉冲。
cfg_ext_register_number	输出	10	配置扩展寄存器编号 读取或写入的配置寄存器的 10 位地址。当 <code>cfg_ext_read_received</code> 或 <code>cfg_ext_write_received</code> 为高电平时，数据有效。
cfg_ext_function_number	输出	8	配置扩展功能编号 对应于配置读取或写入请求的 8 位功能编号。当 <code>cfg_ext_read_received</code> 或 <code>cfg_ext_write_received</code> 为高电平时，数据有效。
cfg_ext_write_data	输出	32	配置扩展写入数据 写入配置寄存器的数据。当 <code>cfg_ext_write_received</code> 为高电平时，此输出有效。
cfg_ext_write_byte_enable	输出	4	针对配置写入传输事务启用“配置扩展写入字节使能字节 (Configuration Extend Write Byte Enable Byte)”。

表 33：配置扩展接口端口描述 (续)

端口	方向	宽度	描述
cfg_ext_read_data	输入	32	配置扩展读取数据 您可通过此总线将数据从外部实现的配置寄存器提供给核。如果您已设置 cfg_ext_read_data_valid，那么核会在将 cfg_ext_read_received 设置为高电平后，在时钟的下一个上升沿上对此数据进行采样。
cfg_ext_read_data_valid	输入	1	配置扩展读取数据有效 用户应用通过向核断言此输入有效，以提供来自外部实现的配置寄存器的数据。核会在将 cfg_ext_read_received 设置为高电平后，在时钟的下一个上升沿对此输入进行采样。

## 配置管理接口端口

配置管理接口用于在配置空间寄存器上执行读取和写入。下表定义了核的配置管理 (Configuration Management) 接口中的端口。

表 34：配置管理接口端口

端口	方向	宽度	描述
cfg_mgmt_addr	输入	19	读取/写入地址。 配置空间 Dword 对齐地址
cfg_mgmt_byte_enable	输入	4	字节使能 针对写入数据的字节使能，其中 cfg_mgmt_byte_enable[0] 对应于 cfg_mgmt_write_data[7:0]，以此类推
cfg_mgmt_read_data	输出	32	读取数据输出 读取数据用于提供配置和管理寄存器的配置
cfg_mgmt_read	输入	1	读取使能 针对读取操作断言有效。高电平有效
cfg_mgmt_read_write_done	输出	1	读取/写入操作完成 当操作完成时，断言有效并保持 1 个周期。高电平有效
cfg_mgmt_write_data	输入	32	写入数据 写入数据用于对配置和管理寄存器进行配置
cfg_mgmt_write	输入	1	写入使能 针对写入操作断言有效。高电平有效

## 描述符旁路模式

如果在 Vivado IDE 的“PCIe DMA”选项卡下，选中“Descriptor Bypass for Read (H2C)”或“Descriptor Bypass for Write (C2H)”，那么这些端口均存在。每个二进制位对应 1 条通道：LSB 对应于通道 0。位元位置中值为 1 表示启用对应的通道描述符旁路。

表 35：H2C 0-3 描述符旁路端口

端口	方向	描述
h2c_dsc_byp_ready	输出	通道已准备好接受新的描述符。当 h2c_dsc_byp_ready 断言无效后，可额外写入 1 个描述符。控制寄存器“运行”位必须断言有效，随后此通道才能接受描述符。

表 35：H2C 0-3 描述符旁路端口 (续)

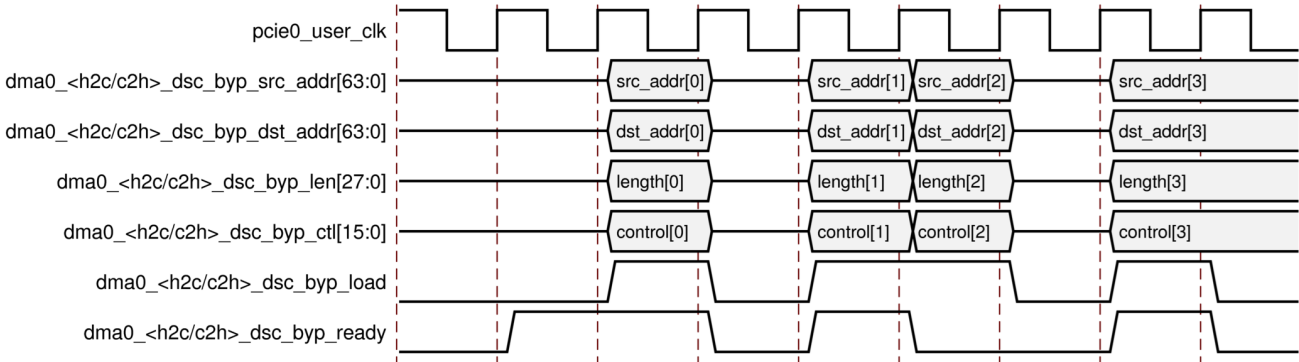
端口	方向	描述
h2c_dsc_byp_load	输入	将 h2c_dsc_byp_data 处存在的描述符写入通道的描述符缓冲器。
h2c_dsc_byp_src_addr[63:0]	输入	要加载的描述符源地址。
h2c_dsc_byp_dst_addr[63:0]	输入	要加载的描述符目标地址。
h2c_dsc_byp_len[27:0]	输入	要加载的描述符长度。
h2c_dsc_byp_ctl[15:0]	输入	要加载的描述符控制。 [0]：停止。设为 1 即可停止提取下一个描述符。 [1]：已完成。设为 1 即可在引擎完成此描述符后中断。 [3:2]：保留。 [4]：EOP。针对 AXI-Stream 接口的包结束。 [15:5]：保留。 所有保留的位均可强制设为 0。

表 36：C2H 0-3 描述符旁路端口

端口	方向	描述
c2h_dsc_byp_ready	输出	通道已准备好接受新的描述符。当 c2h_dsc_byp_ready 断言无效后，可额外写入 1 个描述符。控制寄存器“运行”位必须断言有效，随后此通道才能接受描述符。
c2h_dsc_byp_load	输入	c2h_dsc_byp_* 处存在的描述符有效。
c2h_dsc_byp_src_addr[63:0]	输入	要加载的描述符源地址。
c2h_dsc_byp_dst_addr[63:0]	输入	要加载的描述符目标地址。
c2h_dsc_byp_len[27:0]	输入	要加载的描述符长度。
c2h_dsc_byp_ctl[15:0]	输入	要加载的描述符控制。 [0]：停止。设为 1 即可停止提取下一个描述符。 [1]：已完成。设为 1 即可在引擎完成此描述符后中断。 [3:2]：保留。 [4]：EOP。针对 AXI-Stream 接口的包结束。 [15:5]：保留。 所有保留的位均可强制设为 0。

以下时序图显示了如何在描述符旁路模式下输入描述符。当 dsc\_byp\_ready 断言有效时，即可随 dsc\_byp\_load 信号一并推入新的描述符。

图 8：描述符旁路模式的时序图



**重要提示！** 当 `dsc_byp_ready` 断言无效后，即可立即推入另一个描述符。在以上时序图中，当 `dsc_byp_ready` 断言无效时推入了 1 个描述符。

相关信息

[“PCIe DMA” 选项卡](#)

# 寄存器空间

**注释：** 本文档仅涵盖 DMA 模式寄存器空间。对于 AXI Bridge 模式，请参阅《AXI Bridge for PCI Express Gen3 Subsystem 产品指南》(PG194)。

通过将读取请求或写入请求映射到基址寄存器 (BAR) 即可从主机访问 DMA/Bridge Subsystem for PCI Express® 和用户内部的配置寄存器和状态寄存器。基于 BAR 命中结果，请求将被布线至相应位置。如需了解 PCIe BAR 分配，请参阅[目标桥接器](#)。

## PCIe 到 AXI Bridge 主接口地址映射

到达 PCIe 到 AXI Bridge 主接口的传输事务将被布线至 AXI4 Memory Mapped 用户接口。

## PCIe 到 DMA 地址映射

到达 PCIe 到 DMA 空间的传输事务将被布线至 DMA/Bridge Subsystem for PCI Express® 内部配置寄存器总线。该总线支持 32 位地址空间和 32 位读写请求。

DMA/Bridge Subsystem for PCIe 寄存器可从主机或 AXI 从接口进行访问。这些寄存器应用于 DMA 编程和状态检查。

## PCIe 到 DMA 地址格式

表 37：PCIe 到 DMA 地址格式

31:16	15:12	11:8	7:0
保留	目标	通道	字节偏移

表 38：PCIe 到 DMA 地址字段描述

位索引	字段	描述
15:12	Target	DMA 中的目标子模块 4'h0: H2C 通道 4'h1: C2H 通道 4'h2: IRQ 块 4'h3: 配置 4'h4: H2C SGDMA 4'h5: C2H SGDMA 4'h6: SGDMA 公用 4'h8: MSI-X
11:8	Channel ID[3:0]	该字段仅适用于 H2C 通道、C2H 通道、H2C SGDMA 和 C2H SGDMA 目标。该字段用于表示正在对哪个引擎进行寻址以供这些目标使用。对于所有其它目标，该字段必须为 0。
7:0	Byte Offset	将在目标内访问的寄存器的字节地址。位 [1:0] 必须为 0。

## PCIe 到 DMA 配置寄存器

表 39：配置寄存器属性定义

属性	描述
RV	保留
RW	读取/写入
RC	读取时清除。
W1C	写入 1 以清除
W1S	写入 1 以置位
RO	只读
WO	只写

部分寄存器可通过不同属性来访问。在此类情况下，针对每个属性提供不同的寄存器偏移。未定义的位元和地址空间都将保留。在某些寄存器中，矢量中的个别位元可能表示特定 DMA 引擎。在此类情况下，矢量的 LSB 对应于 H2C 通道（如有）。通道 ID 0 位于 LSB 位置中。表示 C2H 通道的位元直接在这些对应通道的上方进行打包。

## H2C 通道寄存器 (0x0)

本节描述了 H2C 通道寄存器空间。

表 40：H2C 通道寄存器空间

地址（十六进制）	寄存器名称
0x00	H2C Channel Identifier (0x00)
0x04	H2C Channel Control (0x04)
0x08	H2C Channel Control (0x08)
0x0C	H2C Channel Control (0x0C)
0x40	H2C Channel Status (0x40)
0x44	H2C Channel Status (0x44)

表 40：H2C 通道寄存器空间 (续)

地址 (十六进制)	寄存器名称
0x48	H2C Channel Completed Descriptor Count (0x48)
0x4C	H2C Channel Alignments (0x4C)
0x88	H2C Poll Mode Low Write Back Address (0x88)
0x8C	H2C Poll Mode High Write Back Address (0x8C)
0x90	H2C Channel Interrupt Enable Mask (0x90)
0x94	H2C Channel Interrupt Enable Mask (0x94)
0x98	H2C Channel Interrupt Enable Mask (0x98)
0xC0	H2C Channel Performance Monitor Control (0xC0)
0xC4	H2C Channel Performance Cycle Count (0xC4)
0xC8	H2C Channel Performance Cycle Count (0xC8)
0xCC	H2C Channel Performance Data Count (0xCC)
0xD0	H2C Channel Performance Data Count (0xD0)

## H2C Channel Identifier (0x00)

表 41：H2C Channel Identifier (0x00)

位索引	默认值	访问类型	描述
31:20	12'h1fc	RO	子系统标识符
19:16	4'h0	RO	H2C 通道目标
15	1'b0	RO	数据流传输 1: AXI4-Stream 接口 0: AXI4 Memory Mapped 接口
14:12	0	RO	保留
11:8	可变	RO	通道 ID 目标 [3:0]
7:0	8'h04	RO	版本 8'h01: 2015.3 和 2015.4 8'h02: 2016.1 8'h03: 2016.2 8'h04: 2016.3 8'h05: 2016.4 8'h06: 从 2017.1 到最新版本

## H2C Channel Control (0x04)

表 42：H2C Channel Control (0x04)

位索引	默认值	访问类型	描述
31:28			保留
27	1'b0	RW	在 AXI-Stream 模式下禁用 C2H 的置位写回信息时，启用默认写回。

表 42: H2C Channel Control (0x04) (续)

位索引	默认值	访问类型	描述
26	0x0	RW	pollmode_wb_enable 轮询模式写回使能位。 当该位置位后，如果描述符完成时“已完成 (Completed)”位已置位，那么 DMA 就会将已完成的描述符计数写回。
25	1'b0	RW	non_inc_mode 非增量地址模式。仅适用于 m_axi_araddr 接口。
24			保留
23:19	5'h0	RW	ie_desc_error 全部设置为 1 值 (0x1F) 即可启用 Status.Desc_error 记录功能，并在检测到错误时停止引擎。
18:14	5'h0	RW	ie_write_error 全部设置为 1 值 (0x1F) 即可启用 Status.Write_error 记录功能，并在检测到错误时停止引擎。
13:9	5'h0	RW	ie_read_error 全部设置为 1 值 (0x1F) 即可启用 Status.Read_error 记录功能，并在检测到错误时停止引擎。
8:7			保留
6	1'b0	RW	ie_idle_stopped 设为 1 即可启用 Status.Idle_stopped 记录功能
5	1'b0	RW	ie_invalid_length 设为 1 即可启用 Status.Invalid_length 记录功能
4	1'b0	RW	ie_magic_stopped 设为 1 即可启用 Status.Magic_stopped 记录功能
3	1'b0	RW	ie_align_mismatch 设为 1 即可启用 Status.Align_mismatch 记录功能
2	1'b0	RW	ie_descriptor_completed 设为 1 即可启用 Status.Descriptor_completed 记录功能
1	1'b0	RW	ie_descriptor_stopped 设为 1 即可启用 Status.Descriptor_stopped 记录功能
0	1'b0	RW	运行 设为 1 即可启动 SGDMA 引擎。复位为 0 即可停止传输，如果引擎繁忙，那么它会完成当前描述符。

**注释：**

- ie\_\* register 位为中断使能位。当这些位已置位并满足对应条件后，将在 [H2C Channel Status \(0x40\)](#) 中记录状态。正确设置中断掩码（根据 [H2C Channel Interrupt Enable Mask \(0x90\)](#)）后，将生成中断。

## H2C Channel Control (0x08)

表 43: H2C Channel Control (0x08)

位索引	默认值	访问类型	描述
31:28			保留
27:0		W1S	Control 位元描述与 H2C Channel Control (0x04) 中的位元描述相同。



## H2C Channel Control (0x0C)

表 44: H2C Channel Control (0x0C)

位索引	默认值	访问类型	描述
27:0		W1C	Control 位元描述与 H2C Channel Control (0x04) 中的位元描述相同。

## H2C Channel Status (0x40)

表 45: H2C Channel Status (0x40)

位索引	默认值	访问类型	描述
31:24			保留
23:19	5'h0	RW1C	descr_error[4:0] 复位 (0) 条件为：在控制寄存器的运行位进行置位时复位。 4：意外完成 3：报头 EP 2：奇偶校验错误 1：完成方异常中止 0：请求不受支持
18:14	5'h0	RW1C	write_error[4:0] 复位 (0) 条件为：在控制寄存器的运行位进行置位时复位。 位元位置： 4-2：保留 1：从接口错误 0：解码器错误
13:9	5'h0	RW1C	read_error[4:0] 复位 (0) 条件为：在控制寄存器的运行位进行置位时复位。 位元位置 4：意外完成 3：报头 EP 2：奇偶校验错误 1：完成方异常中止 0：请求不受支持
8:7			保留
6	1'b0	RW1C	idle_stopped 复位 (0) 条件为：在控制寄存器的运行位进行置位时复位。置位条件为：在控制寄存器 ie_idle_stopped 位已置位的前提下，将控制寄存器运行位复位之后，引擎处于空闲状态时进行置位。
5	1'b0	RW1C	invalid_length 复位条件为：在控制寄存器运行位进行置位时复位。置位条件为：当描述符长度并非 AXI4-Stream 通道的数据宽度的倍数，并且控制寄存器的 ie_invalid_length 位已置位时进行置位。
4	1'b0	RW1C	magic_stopped 复位条件为：在控制寄存器运行位进行置位时复位。置位条件为：在控制寄存器 ie_magic_stopped 位已置位的前提下，当引擎遇到含无效 magic 的描述符并停止时进行置位。
3	1'b0	RW1C	align_mismatch 描述符上的源地址与目标地址彼此未正确对齐。

表 45: H2C Channel Status (0x40) (续)

位索引	默认值	访问类型	描述
2	1'b0	RW1C	descriptor_completed 复位条件为：在控制寄存器运行位进行置位时复位。置位条件为：在控制寄存器 ie_descriptor_stopped 位已置位的前提下，当引擎已完成描述符并且 COMPLETE 位已置位时进行置位。
1	1'b0	RW1C	descriptor_stopped 复位条件为：在控制寄存器运行位进行置位时复位。置位条件为：在控制寄存器 ie_descriptor_stopped 位已置位的前提下，当引擎已完成描述符并且 STOP 位已置位时进行置位。
0	1'b0	RO	Busy 置位条件为：当 SGDMA 引擎处于繁忙状态时置位。当该引擎处于空闲状态时为零 (0)。

## H2C Channel Status (0x44)

表 46: H2C Channel Status (0x44)

位索引	默认值	访问类型	描述
23:1		RC	状态 读取时清除。位元描述与 H2C Channel Status (0x40) 中的位元描述相同。 位 0 无法清除。

## H2C Channel Completed Descriptor Count (0x48)

表 47: H2C Channel Completed Descriptor Count (0x48)

位索引	默认值	访问类型	描述
31:0	32'h0	RO	compl_descriptor_count 表示完成列表中每个描述符后，引擎完成的描述符更新数量。在控制寄存器的上升沿复位为 0，该位元为运行位 (H2C Channel Control (0x04))。

## H2C Channel Alignments (0x4C)

表 48: H2C Channel Alignments (0x4C)

位索引	默认值	访问类型	描述
23:16	基于配置	RO	addr_alignment 源地址和目标地址必须对齐到此字节对齐。该值取决于配置参数。
15:8	基于配置	RO	len_granularity DMA 传输的最小粒度为字节。
7:0	基于配置	RO	address_bits 已配置的地址位的数量。

## H2C Poll Mode Low Write Back Address (0x88)

表 49: H2C Poll Mode Low Write Back Address (0x88)

位索引	默认值	访问类型	描述
31:0	0x0	RW	Pollmode_lo_wb_addr[31:0] 轮询模式写回地址的下 32 位。

## H2C Poll Mode High Write Back Address (0x8C)

表 50: H2C Poll Mode High Write Back Address (0x8C)

位索引	默认值	访问类型	描述
31:0	0x0	RW	Pollmode_hi_wb_addr[63:32] 轮询模式写回地址的上 32 位。

## H2C Channel Interrupt Enable Mask (0x90)

表 51: H2C Channel Interrupt Enable Mask (0x90)

位索引	默认值	访问类型	描述
23:19	5'h0	RW	im_desc_error[4:0] 设为 1 则会在记录对应的状态寄存器 read_error 位时发生中断。
18:14	5'h0	RW	im_write_error[4:0] 设为 1 则会在记录对应的状态寄存器 write_error 位时发生中断。
13:9	5'h0	RW	im_read_error[4:0] 设为 1 则会在记录对应的状态寄存器 read_error 位时发生中断。
8:7			保留
6	1'b0	RW	im_idle_stopped 设为 1 则会在记录状态寄存器 idle_stopped 位时发生中断。
5	1'b0	RW	im_invalid_length 设为 1 则会在记录状态寄存器 invalid_length 位时发生中断。
4	1'b0	RW	im_magic_stopped 设为 1 则会在记录状态寄存器 magic_stopped 位时发生中断。
3	1'b0	RW	im_align_mismatch 设为 1 则会在记录状态寄存器 align_mismatch 位时发生中断。
2	1'b0	RW	im_descriptor_completd 设为 1 则会在记录状态寄存器 descriptor_completed 位时发生中断。
1	1'b0	RW	im_descriptor_stopped 设为 1 则会在记录状态寄存器 descriptor_stopped 位时发生中断。

## H2C Channel Interrupt Enable Mask (0x94)

表 52: H2C Channel Interrupt Enable Mask (0x94)

位索引	默认值	访问类型	描述
		W1S	中断使能掩码

## H2C Channel Interrupt Enable Mask (0x98)

表 53: H2C Channel Interrupt Enable Mask (0x98)

位索引	默认值	访问类型	描述
		W1C	中断使能掩码

## H2C Channel Performance Monitor Control (0xC0)

表 54: H2C Channel Performance Monitor Control (0xC0)

位索引	默认值	访问类型	描述
2	1'b0	RW	运行 设为 1 以装备性能计数器。在控制寄存器运行位已置位后，即可启动计数器。 设为 0 以停止性能计数器。
1	1'b0	WO	清除 写入 1 即可清除性能计数器。
0	1'b0	RW	自动 当含停止位的描述符完成后，即自动停止性能计数器。在控制寄存器运行位已置位后，即自动清除性能计数器。要启动计数器，则仍需向性能监控寄存器运行位写入 1。

## H2C Channel Performance Cycle Count (0xC4)

表 55: H2C Channel Performance Cycle Count (0xC4)

位索引	默认值	访问类型	描述
31:0	32'h0	RO	pmon_cyc_count[31:0] 运行时，按每个时钟递增。请参阅 Performance Monitor Control 寄存器 (0xC0) 的“清除”位和“自动”位以了解有关清除的信息。

## H2C Channel Performance Cycle Count (0xC8)

表 56: H2C Channel Performance Cycle Count (0xC8)

位索引	默认值	访问类型	描述
16	1'b0	RO	pmon_cyc_count_maxed 已达周期计数最大值。

表 56: H2C Channel Performance Cycle Count (0xC8) (续)

位索引	默认值	访问类型	描述
9:0	10'h0	RO	pmon_cyc_count [41:32] 运行时，按每个时钟递增。请参阅 Performance Monitor Control 寄存器 (0xC0) 的“清除”位和“自动”位以了解有关清除的信息。

## H2C Channel Performance Data Count (0xCC)

表 57: H2C Channel Performance Data Count (0xCC)

位索引	默认值	访问类型	描述
31:0	32'h0	RO	pmon_dat_count[31:0] 运行时，随每个有效读取数据节拍递增。请参阅 Performance Monitor Control 寄存器 (0xC0) 的“清除”位和“自动”位以了解有关清除的信息。

## H2C Channel Performance Data Count (0xD0)

表 58: H2C Channel Performance Data Count (0xD0)

位索引	默认值	访问类型	描述
16	1'b0	RO	pmon_dat_count_maxed 已达数据计数最大值
15:10			保留
9:0	10'h0	RO	pmon_dat_count [41:32] 运行时，随每个有效读取数据节拍递增。请参阅 Performance Monitor Control 寄存器 (0xC0) 的“清除”位和“自动”位以了解有关清除的信息。

## C2H 通道寄存器 (0x1)

本节描述了 C2H 通道寄存器空间。

表 59: C2H 通道寄存器空间

地址 (十六进制)	寄存器名称
0x00	C2H Channel Identifier (0x00)
0x04	C2H Channel Control (0x04)
0x08	C2H Channel Control (0x08)
0x0C	C2H Channel Control (0x0C)
0x40	C2H Channel Status (0x40)
0x44	C2H Channel Status (0x44)
0x48	C2H Channel Completed Descriptor Count (0x48)
0x4C	C2H Channel Alignments (0x4C)
0x88	C2H Poll Mode Low Write Back Address (0x88)
0x8C	C2H Poll Mode High Write Back Address (0x8C)

表 59: C2H 通道寄存器空间 (续)

地址 (十六进制)	寄存器名称
0x90	C2H Channel Interrupt Enable Mask (0x90)
0x94	C2H Channel Interrupt Enable Mask (0x94)
0x98	C2H Channel Interrupt Enable Mask (0x98)
0xC0	C2H Channel Performance Monitor Control (0xC0)
0xC4	C2H Channel Performance Cycle Count (0xC4)
0xC8	C2H Channel Performance Cycle Count (0xC8)
0xCC	C2H Channel Performance Data Count (0xCC)
0xD0	C2H Channel Performance Data Count (0xD0)

## C2H Channel Identifier (0x00)

表 60: C2H Channel Identifier (0x00)

位索引	默认值	访问类型	描述
31:20	12'h1fc	RO	子系统标识符
19:16	4'h1	RO	C2H 通道目标
15	1'b0	RO	数据流传输 1: AXI4-Stream 接口 0: AXI4 Memory Mapped 接口
14:12	0	RO	保留
11:8	可变	RO	通道 ID 目标 [3:0]
7:0	8'h04	RO	版本 8'h01: 2015.3 和 2015.4 8'h02: 2016.1 8'h03: 2016.2 8'h04: 2016.3 8'h05: 2016.4 8'h06: 从 2017.1 到最新版本

## C2H Channel Control (0x04)

表 61: C2H Channel Control (0x04)

位索引	默认值	访问类型	描述
31:28			保留
27	0x0	RW	为 C2H AXI4-Stream 禁用元数据写回。如果此通道配置为使用 AXI Memory Mapped, 则无任何影响。
26	0x0	RW	pollmode_wb_enable 轮询模式写回使能位。 当该位置位后, 如果描述符完成时“已完成 (Completed)”位已置位, 那么 DMA 就会将已完成的描述符计数写回。
25	1'b0	RW	non_inc_mode 非增量地址模式。仅适用于 m_axi_araddr 接口。

表 61：C2H Channel Control (0x04) (续)

位索引	默认值	访问类型	描述
23:19	5'h0	RW	ie_desc_error 全部设置为 1 值 (0x1F) 即可启用 Status.Desc_error 记录功能，并在检测到错误时停止引擎。
13:9	5'h0	RW	ie_read_error 全部设置为 1 值 (0x1F) 即可启用 Status.Read_error 记录功能，并在检测到错误时停止引擎
8:7			保留
6	1'b0	RW	ie_idle_stopped 设为 1 即可启用 Status.Idle_stopped 记录功能
5	1'b0	RW	ie_invalid_length 设为 1 即可启用 Status.Invalid_length 记录功能
4	1'b0	RW	ie_magic_stopped 设为 1 即可启用 Status.Magic_stopped 记录功能
3	1'b0	RW	ie_align_mismatch 设为 1 即可启用 Status.Align_mismatch 记录功能
2	1'b0	RW	ie_descriptor_completed 设为 1 即可启用 Status.Descriptor_completed 记录功能
1	1'b0	RW	ie_descriptor_stopped 设为 1 即可启用 Status.Descriptor_stopped 记录功能
0	1'b0	RW	运行 设为 1 即可启动 SGDMA 引擎。复位为 0 即可停止传输，如果引擎繁忙，那么它会完成当前描述符。

**注释：**

- ie\_\* register 位为中断使能位。当这些位已置位并满足对应条件后，将在 [C2H Channel Status \(0x40\)](#) 中记录状态。正确设置中断掩码（根据 [C2H Channel Interrupt Enable Mask \(0x90\)](#)）后，将生成中断。

## C2H Channel Control (0x08)

表 62：C2H Channel Control (0x08)

位索引	默认值	访问类型	描述
		W1S	Control 位元描述与 C2H Channel Control (0x04) 中的位元描述相同。

## C2H Channel Control (0x0C)

表 63：C2H Channel Control (0x0C)

位索引	默认值	访问类型	描述
		W1C	Control 位元描述与 C2H Channel Control (0x04) 中的位元描述相同。

## C2H Channel Status (0x40)

表 64：C2H Channel Status (0x40)

位索引	默认值	访问类型	描述
23:19	5'h0	RW1C	descr_error[4:0] 复位 (0) 条件为：在控制寄存器的运行位进行置位时复位。 位元位置： 4：意外完成 3：报头 EP 2：奇偶校验错误 1：完成方异常中止 0：请求不受支持
13:9	5'h0	RW1C	read_error[4:0] 复位 (0) 条件为：在控制寄存器的运行位进行置位时复位。 位元位置： 4-2：保留 1：从接口错误 0：解码器错误
8:7			保留
6	1'b0	RW1C	idle_stopped 复位 (0) 条件为：在控制寄存器的运行位进行置位时复位。置位条件为：在控制寄存器 ie_idle_stopped 位已置位的前提下，将控制寄存器运行位复位之后，引擎处于空闲状态时进行置位。
5	1'b0	RW1C	invalid_length 复位条件为：在控制寄存器运行位进行置位时复位。置位条件为：当描述符长度并非 AXI4-Stream 通道的数据宽度的倍数，并且控制寄存器的 ie_invalid_length 位已置位时进行置位。
4	1'b0	RW1C	magic_stopped 复位条件为：在控制寄存器运行位进行置位时复位。置位条件为：在控制寄存器 ie_magic_stopped 位已置位的前提下，当引擎遇到含无效 magic 的描述符并停止时进行置位。
3	13'b0	RW1C	align_mismatch 描述符上的源地址与目标地址彼此未正确对齐。
2	1'b0	RW1C	descriptor_completed 复位条件为：在控制寄存器运行位进行置位时复位。置位条件为：在控制寄存器 ie_descriptor_completed 位已置位的前提下，当引擎已完成描述符并且 COMPLETE 位已置位时进行置位。
1	1'b0	RW1C	descriptor_stopped 复位条件为：在控制寄存器运行位进行置位时复位。置位条件为：在控制寄存器 ie_magic_stopped 位已置位的前提下，当引擎已完成描述符并且 STOP 位已置位时进行置位。
0	1'b0	RO	Busy 置位条件为：当 SGDMA 引擎处于繁忙状态时置位。当该引擎处于空闲状态时为零 (0)。



**C2H Channel Status (0x44)**

表 65: C2H Channel Status (0x44)

位索引	默认值	访问类型	描述
23:1		RC	状态 位元描述与 C2H Channel Status (0x40) 中的位元描述相同。

**C2H Channel Completed Descriptor Count (0x48)**

表 66: C2H Channel Completed Descriptor Count (0x48)

位索引	默认值	访问类型	描述
31:0	32'h0	RO	compl_descriptor_count 表示完成列表中每个描述符后，引擎完成的描述符更新数量。 在控制寄存器的上升沿复位为 0，该位元为运行位 (C2H Channel Control (0x04))。

**C2H Channel Alignments (0x4C)**

表 67: C2H Channel Alignments (0x4C)

位索引	默认值	访问类型	描述
23:16	可变	RO	addr_alignment 源地址和目标地址必须对齐到此字节对齐。该值取决于配置参数。
15:8	可变	RO	len_granularity DMA 传输的最小粒度为字节。
7:0	ADDR_BITS	RO	address_bits 已配置的地址位的数量。

**C2H Poll Mode Low Write Back Address (0x88)**

表 68: C2H Poll Mode Low Write Back Address (0x88)

位索引	默认值	访问类型	描述
31:0	0x0	RW	Pollmode_lo_wb_addr[31:0] 轮询模式写回地址的下 32 位。

**C2H Poll Mode High Write Back Address (0x8C)**

表 69: C2H Poll Mode High Write Back Address (0x8C)

位索引	默认值	访问类型	描述
31:0	0x0	RW	Pollmode_hi_wb_addr[63:32] 轮询模式写回地址的上 32 位。

## C2H Channel Interrupt Enable Mask (0x90)

表 70：C2H Channel Interrupt Enable Mask (0x90)

位索引	默认值	访问类型	描述
23:19	5'h0	RW	im_desc_error[4:0] 设为 1 则会在记录对应的 Status.Read_Error 时发生中断。
13:9	5'h0	RW	im_read_error[4:0] 设为 1 则会在记录对应的 Status.Read_Error 时发生中断。
8:7			保留
6	1'b0	RW	im_idle_stopped 设为 1 则会在记录 Status.Idle_stopped 时发生中断。
4	1'b0	RW	im_magic_stopped 设为 1 则会在记录 Status.Magic_stopped 时发生中断。
2	1'b0	RW	im_descriptor_completd 设为 1 则会在记录 Status.Descriptor_completed 时发生中断。
1	1'b0	RW	im_descriptor_stopped 设为 1 则会在记录 Status.Descriptor_stopped 时发生中断。
0			保留

## C2H Channel Interrupt Enable Mask (0x94)

表 71：C2H Channel Interrupt Enable Mask (0x94)

位索引	默认值	访问类型	描述
		W1S	中断使能掩码 位元描述与 C2H Channel Interrupt Enable Mask (0x90) 中的位元描述相同。

## C2H Channel Interrupt Enable Mask (0x98)

表 72：C2H Channel Interrupt Enable Mask (0x98)

位索引	默认值	访问类型	描述
		W1C	中断使能掩码 位元描述与 C2H Channel Interrupt Enable Mask (0x90) 中的位元描述相同。

## C2H Channel Performance Monitor Control (0xC0)

表 73：C2H Channel Performance Monitor Control (0xC0)

位索引	默认值	访问类型	描述
2	1'b0	RW	运行 设为 1 以装备性能计数器。在控制寄存器运行位已置位后，即可启动计数器。 设为 0 以停止性能计数器。

表 73: C2H Channel Performance Monitor Control (0xC0) (续)

位索引	默认值	访问类型	描述
1	1'b0	WO	清除 写入 1 即可清除性能计数器。
0	1'b0	RW	自动 当含停止位的描述符完成后，即自动停止性能计数器。在控制寄存器运行位已置位后，即自动清除性能计数器。要启动计数器，则仍需向性能监控寄存器运行位写入 1。

## C2H Channel Performance Cycle Count (0xC4)

表 74: C2H Channel Performance Cycle Count (0xC4)

位索引	默认值	访问类型	描述
31:0	32'h0	RO	pmon_cyc_count[31:0] 运行时，按每个时钟递增。请参阅 Performance Monitor Control 寄存器 (0xC0) 的“清除”位和“自动”位以了解有关清除的信息。

## C2H Channel Performance Cycle Count (0xC8)

表 75: C2H Channel Performance Cycle Count (0xC8)

位索引	默认值	访问类型	描述
16	1'b0	RO	pmon_cyc_count_maxed 已达周期计数最大值。
15:10			保留
9:0	10'h0	RO	pmon_cyc_count [41:32] 运行时，按每个时钟递增。请参阅 Performance Monitor Control 寄存器 (0xC0) 的“清除”位和“自动”位以了解有关清除的信息。

## C2H Channel Performance Data Count (0xCC)

表 76: C2H Channel Performance Data Count (0xCC)

位索引	默认值	访问类型	描述
31:0	32'h0	RO	pmon_dat_count[31:0] 运行时，随每个有效读取数据节拍递增。请参阅 Performance Monitor Control 寄存器 (0xC0) 的“清除”位和“自动”位以了解有关清除的信息。

## C2H Channel Performance Data Count (0xD0)

表 77: C2H Channel Performance Data Count (0xD0)

位索引	默认值	访问类型	描述
16	1'b0	RO	pmon_dat_count_maxed 已达数据计数最大值

表 77: C2H Channel Performance Data Count (0xD0) (续)

位索引	默认值	访问类型	描述
15:10			保留
9:0	10'h0	RO	pmon_dat_count [41:32] 运行时，随每个有效读取数据节拍递增。请参阅 Performance Monitor Control 寄存器 (0xC0) 的“清除”位和“自动”位以了解有关清除的信息。

## IRQ 块寄存器 (0x2)

本节描述了 IRQ 块寄存器。

表 78: IRQ 块寄存器空间

地址 (十六进制)	寄存器名称
0x00	IRQ Block Identifier (0x00)
0x04	IRQ Block User Interrupt Enable Mask (0x04)
0x08	IRQ Block User Interrupt Enable Mask (0x08)
0x0C	IRQ Block User Interrupt Enable Mask (0x0C)
0x10	IRQ Block Channel Interrupt Enable Mask (0x10)
0x14	IRQ Block Channel Interrupt Enable Mask (0x14)
0x18	IRQ Block Channel Interrupt Enable Mask (0x18)
0x40	IRQ Block User Interrupt Request (0x40)
0x44	IRQ Block Channel Interrupt Request (0x44)
0x48	IRQ Block User Interrupt Pending (0x48)
0x4C	IRQ Block Channel Interrupt Pending (0x4C)
0x80	IRQ Block User Vector Number (0x80)
0x84	IRQ Block User Vector Number (0x84)
0x88	IRQ Block User Vector Number (0x88)
0x8C	IRQ Block User Vector Number (0x8C)
0xA0	IRQ Block Channel Vector Number (0xA0)
0xA4	IRQ Block Channel Vector Number (0xA4)

AXI Bridge 与 AXI DMA 共享中断处理寄存器。在 AXI Bridge 模式下选中“MSI-X Capabilities”时，来自 BAR0 的 64 KB 地址空间将保留以供 MSI-X 表格使用。默认情况下，寄存器空间在 BAR0 内分配。您可通过使用 `CONFIG.bar_indicator {BAR0} Tci` 命令来选择其它 BAR 中的寄存器空间（从 BAR1 到 BAR5）。仅当选中“MSI-X Capabilities”选项时，该选项才可用。对于其它中断选项，则不分配任何空间。

## IRQ Block Identifier (0x00)

表 79: IRQ Block Identifier (0x00)

位索引	默认值	访问类型	描述
31:20	12'h1fc	RO	子系统标识符
19:16	4'h2	RO	IRQ 标识符

表 79: IRQ Block Identifier (0x00) (续)

位索引	默认值	访问类型	描述
15:8	8'h0	RO	保留
7:0	8'h04	RO	版本 8'h01: 2015.3 和 2015.4 8'h02: 2016.1 8'h03: 2016.2 8'h04: 2016.3 8'h05: 2016.4 8'h06: 从 2017.1 到最新版本

### IRQ Block User Interrupt Enable Mask (0x04)

表 80: IRQ Block User Interrupt Enable Mask (0x04)

位索引	默认值	访问类型	描述
[NUM_USR_INT-1:0]	'h0	RW	user_int_enmask 用户中断使能掩码 0: 当用户中断源断言有效时，阻止生成中断。 1: 在用户中断源的上升沿生成中断。如果“使能掩码 (Enable Mask)”已置位，并且源也已置位，那么同样会生成用户中断。

### IRQ Block User Interrupt Enable Mask (0x08)

表 81: IRQ Block User Interrupt Enable Mask (0x08)

位索引	默认值	访问类型	描述
		W1S	user_int_enmask 位元描述与 IRQ Block User Interrupt Enable Mask (0x04) 中的位元描述相同。

### IRQ Block User Interrupt Enable Mask (0x0C)

表 82: IRQ Block User Interrupt Enable Mask (0x0C)

位索引	默认值	访问类型	描述
		W1C	user_int_enmask 位元描述与 IRQ Block User Interrupt Enable Mask (0x04) 中的位元描述相同。

IRQ Block Channel Interrupt Enable Mask (0x10)

表 83: IRQ Block Channel Interrupt Enable Mask (0x10)

位索引	默认值	访问类型	描述
[NUM_CHNL-1:0]	'h0	RW	channel_int_enmask 引擎中断使能掩码。每个读取或写入引擎对应 1 个位。 0: 中断源断言有效时，阻止生成中断。H2C 位元的位置始终从位 0 开始。C2H 位元的位置为高于最后一个 H2C 索引的索引，因此它取决于 NUM_H2C_CHNL 参数。 1: 在中断源的上升沿生成中断。如果 enmask 位已置位并且源已置位，那么同样会生成中断。

IRQ Block Channel Interrupt Enable Mask (0x14)

表 84: IRQ Block Channel Interrupt Enable Mask (0x14)

位索引	默认值	访问类型	描述
		W1S	channel_int_enmask 位元描述与 IRQ Block Channel Interrupt Enable Mask (0x10) 中的位元描述相同。

IRQ Block Channel Interrupt Enable Mask (0x18)

表 85: IRQ Block Channel Interrupt Enable Mask (0x18)

位索引	默认值	访问类型	描述
		W1C	channel_int_enmask 位元描述与 IRQ Block Channel Interrupt Enable Mask (0x10) 中的位元描述相同。

下图显示了 H2C 位元与 C2H 位元的打包方式。

图 9: 打包 H2C 和 C2H

位	7	6	5	4	3	2	1	0
启用 4 个 H2C 和 4 个 C2H	C2H_3	C2H_2	C2H_1	C2H_0	H2C_3	H2C_2	H2C_1	H2C_0
启用 3 个 H2C 和 3 个 C2H	X	X	C2H_2	C2H_1	C2H_0	H2C_2	H2C_1	H2C_0
启用 1 个 H2C 和 3 个 C2H	X	X	X	X	C2H_2	C2H_1	C2H_0	H2C_0

X15954-051621

## IRQ Block User Interrupt Request (0x40)

表 86：IRQ Block User Interrupt Request (0x40)

位索引	默认值	访问类型	描述
[NUM_USR_INT-1:0]	'h0	RO	user_int_req 用户中断请求 此寄存器反映仅当中断源与使能掩码寄存器同时断言有效时，此中断位才会断言有效。

## IRQ Block Channel Interrupt Request (0x44)

表 87：IRQ Block Channel Interrupt Request (0x44)

位索引	默认值	访问类型	描述
[NUM_CHNL-1:0]	'h0	RO	engine_int_req 引擎中断请求。每个读取或写入引擎对应 1 个位。此寄存器反映仅当中断源与使能掩码寄存器同时断言有效时，此中断位才会断言有效。H2C 位元的位置始终从位 0 开始。C2H 位元的位置为高于最后一个 H2C 索引的索引，因此它取决于 NUM_H2C_CHNL 参数。上图显示了 H2C 位元与 C2H 位元的打包方式。

## IRQ Block User Interrupt Pending (0x48)

表 88：IRQ Block User Interrupt Pending (0x48)

位索引	默认值	访问类型	描述
[NUM_USR_INT-1:0]	'h0	RO	user_int_pend 用户中断暂挂。 此寄存器表示存在暂挂事件。通过移除位于源组件上的事件原因条件即可清除暂挂事件。

## IRQ Block Channel Interrupt Pending (0x4C)

表 89：IRQ Block Channel Interrupt Pending (0x4C)

位索引	默认值	访问类型	描述
[NUM_CHNL-1:0]	'h0	RO	engine_int_pend 引擎中断暂挂。 每个读取或写入引擎对应 1 个位。此寄存器表示存在暂挂事件。通过移除位于源组件上的事件原因条件即可清除暂挂事件。H2C 位元的位置始终从位 0 开始。C2H 位元的位置为高于最后一个 H2C 索引的索引，因此它取决于 NUM_H2C_CHNL 参数。上图显示了 H2C 位元与 C2H 位元的打包方式。

## IRQ Block User Vector Number (0x80)

如果启用 MSI，那么此寄存器会指定 MSI 或 MSI-X 的 MSI-X 矢量编号。在传统中断中，每个字段仅限 2 个 LSB 用于映射到 INTA、B、C 或 D。

表 90: IRQ Block User Vector Number (0x80)

位索引	默认值	访问类型	描述
28:24	5'h0	RW	矢量 3 矢量编号，当用户 IRQ usr_irq_req[3] 生成中断时使用。
20:16	5'h0	RW	矢量 2 矢量编号，当用户 IRQ usr_irq_req[2] 生成中断时使用。
12:8	5'h0	RW	矢量 1 矢量编号，当用户 IRQ usr_irq_req[1] 生成中断时使用。
4:0	5'h0	RW	矢量 0 矢量编号，当用户 IRQ usr_irq_req[0] 生成中断时使用。

### IRQ Block User Vector Number (0x84)

如果启用 MSI，那么此寄存器会指定 MSI 或 MSI-X 的 MSI-X 矢量编号。在传统中断中，每个字段仅限 2 个 LSB 用于映射到 INTA、B、C 或 D。

表 91: IRQ Block User Vector Number (0x84)

位索引	默认值	访问类型	描述
28:24	5'h0	RW	矢量 7 矢量编号，当用户 IRQ usr_irq_req[7] 生成中断时使用。
20:16	5'h0	RW	矢量 6 矢量编号，当用户 IRQ usr_irq_req[6] 生成中断时使用。
12:8	5'h0	RW	矢量 5 矢量编号，当用户 IRQ usr_irq_req[5] 生成中断时使用。
4:0	5'h0	RW	矢量 4 矢量编号，当用户 IRQ usr_irq_req[4] 生成中断时使用。

### IRQ Block User Vector Number (0x88)

如果启用 MSI，那么此寄存器会指定 MSI 或 MSI-X 的 MSI-X 矢量编号。在传统中断中，每个字段仅限 2 个 LSB 用于映射到 INTA、B、C 或 D。

表 92: IRQ Block User Vector Number (0x88)

位索引	默认值	访问类型	描述
28:24	5'h0	RW	矢量 11 矢量编号，当用户 IRQ usr_irq_req[11] 生成中断时使用。
20:16	5'h0	RW	矢量 10 矢量编号，当用户 IRQ usr_irq_req[10] 生成中断时使用。
12:8	5'h0	RW	矢量 9 矢量编号，当用户 IRQ usr_irq_req[9] 生成中断时使用。
4:0	5'h0	RW	矢量 8 矢量编号，当用户 IRQ usr_irq_req[8] 生成中断时使用。



## IRQ Block User Vector Number (0x8C)

如果启用 MSI，那么此寄存器会指定 MSI 或 MSI-X 的 MSI-X 矢量编号。在传统中断中，每个字段仅限 2 个 LSB 用于映射到 INTA、B、C 或 D。

表 93：IRQ Block User Vector Number (0x8C)

位索引	默认值	访问类型	描述
28:24	5'h0	RW	矢量 15 矢量编号，当用户 IRQ usr_irq_req[15] 生成中断时使用。
20:16	5'h0	RW	矢量 14 矢量编号，当用户 IRQ usr_irq_req[14] 生成中断时使用。
12:8	5'h0	RW	矢量 13 矢量编号，当用户 IRQ usr_irq_req[13] 生成中断时使用。
4:0	5'h0	RW	矢量 12 矢量编号，当用户 IRQ usr_irq_req[12] 生成中断时使用。

## IRQ Block Channel Vector Number (0xA0)

如果启用 MSI，那么此寄存器会指定 MSI 的 MSI 矢量编号。在传统中断中，每个字段仅限 2 个 LSB 用于映射到 INTA、B、C 或 D。

与其它 C2H/H2C 位打包澄清信息相似，请参阅上图。第一个 C2H 矢量位于最后一个 H2C 矢量之后。例如，如果 NUM\_H2C\_Channel = 1，那么 H2C0 矢量位于 0xA0（位 [4:0]），而 C2H 通道 0 矢量则位于 0xA0（位 [12:8]）。如果 NUM\_H2C\_Channel = 4，那么 H2C3 矢量位于 0xA0 28:24，而 C2H 通道 0 矢量则位于 0xA4（位 [4:0]）。

表 94：IRQ Block Channel Vector Number (0xA0)

位索引	默认值	访问类型	描述
28:24	5'h0	RW	vector3 矢量编号，当通道 3 生成中断时使用。
20:16	5'h0	RW	vector2 矢量编号，当通道 2 生成中断时使用。
12:8	5'h0	RW	vector1 矢量编号，当通道 1 生成中断时使用。
4:0	5'h0	RW	vector0 矢量编号，当通道 0 生成中断时使用。

## IRQ Block Channel Vector Number (0xA4)

如果启用 MSI，那么此寄存器会指定 MSI 的 MSI 矢量编号。在传统中断中，每个字段仅限 2 个 LSB 用于映射到 INTA、B、C 或 D。

与其它 C2H/H2C 位打包澄清信息相似，请参阅上图。第一个 C2H 矢量位于最后一个 H2C 矢量之后。例如，如果 NUM\_H2C\_Channel = 1，那么 H2C0 矢量位于 0xA0（位 [4:0]），而 C2H 通道 0 矢量则位于 0xA0（位 [12:8]）。如果 NUM\_H2C\_Channel = 4，那么 H2C3 矢量位于 0xA0 28:24，而 C2H 通道 0 矢量则位于 0xA4（位 [4:0]）。

表 95: IRQ Block Channel Vector Number (0xA4)

位索引	默认值	访问类型	描述
28:24	5'h0	RW	vector7 矢量编号，当通道 7 生成中断时使用。
20:16	5'h0	RW	vector6 矢量编号，当通道 6 生成中断时使用。
12:8	5'h0	RW	vector5 矢量编号，当通道 5 生成中断时使用。
4:0	5'h0	RW	vector4 矢量编号，当通道 4 生成中断时使用。

## 配置块寄存器 (0x3)

本节描述了配置块寄存器。

表 96: 配置块寄存器空间

地址 (十六进制)	寄存器名称
0x00	Config Block Identifier (0x00)
0x04	Config Block BusDev (0x04)
0x08	Config Block PCIE Max Payload Size (0x08)
0x0C	Config Block PCIE Max Read Request Size (0x0C)
0x10	Config Block System ID (0x10)
0x14	Config Block MSI Enable (0x14)
0x18	Config Block PCIE Data Width (0x18)
0x1C	Config PCIE Control (0x1C)
0x40	Config AXI User Max Payload Size (0x40)
0x44	Config AXI User Max Read Request Size (0x44)
0x60	Config Write Flush Timeout (0x60)

## Config Block Identifier (0x00)

表 97: Config Block Identifier (0x00)

位索引	默认值	访问类型	描述
31:20	12'h1fc	RO	子系统标识符
19:16	4'h3	RO	配置标识符
15:8	8'h0	RO	保留
7:0	8'h04	RO	版本 8'h01: 2015.3 和 2015.4 8'h02: 2016.1 8'h03: 2016.2 8'h04: 2016.3 8'h05: 2016.4 8'h06: 从 2017.1 到最新版本

## Config Block BusDev (0x04)

表 98: Config Block BusDev (0x04)

位索引	默认值	访问类型	描述
[15:0]	PCIe IP	RO	bus_dev 总线、器件和功能

## Config Block PCIE Max Payload Size (0x08)

表 99: Config Block PCIE Max Payload Size (0x08)

位索引	默认值	访问类型	描述
[2:0]	PCIe IP	RO	pcie_max_payload 最大写入有效载荷大小。该值取 PCIe IP MPS 与 DMA/Bridge Subsystem for PCIe 参数之间的较小的值。 3'b000: 128 个字节 3'b001: 256 个字节 3'b010: 512 个字节 3'b011: 1024 个字节 3'b100: 2048 个字节 3'b101: 4096 个字节

## Config Block PCIE Max Read Request Size (0x0C)

表 100: Config Block PCIE Max Read Request Size (0x0C)

位索引	默认值	访问类型	描述
[2:0]	PCIe IP	RO	pcie_max_read 最大读取请求大小。该值取 PCIe IP MRRS 与 DMA/Bridge Subsystem for PCIe 参数之间的较小的值。 3'b000: 128 个字节 3'b001: 256 个字节 3'b010: 512 个字节 3'b011: 1024 个字节 3'b100: 2048 个字节 3'b101: 4096 个字节

## Config Block System ID (0x10)

表 101: Config Block System ID (0x10)

位索引	默认值	访问类型	描述
[15:0]	16'hff01	RO	system_id 核系统 ID

## Config Block MSI Enable (0x14)

表 102: Config Block MSI Enable (0x14)

位索引	默认值	访问类型	描述
[0]	PCIe IP	RO	MSI_en MSI 使能
[1]	PCIe IP	RO	MSI-X 使能

## Config Block PCIE Data Width (0x18)

表 103: Config Block PCIE Data Width (0x18)

位索引	默认值	访问类型	描述
[2:0]	C_DAT_WIDTH	RO	pcie_width PCIe AXI4-Stream 宽度 0: 64 位 1: 128 位 2: 256 位 3: 512 位

## Config PCIE Control (0x1C)

表 104: Config PCIE Control (0x1C)

位索引	默认值	访问类型	描述
[0]	1'b1	RW	宽松排序 当宽松排序位已置位时，即可生成 PCIe 读取请求 TLP。

## Config AXI User Max Payload Size (0x40)

表 105: Config AXI User Max Payload Size (0x40)

位索引	默认值	访问类型	描述
6:4	3'h5	RO	user_eff_payload 向用户应用发出的实际最大有效载荷大小。由于 IP 配置或数据路径宽度，该值可能低于 user_prg_payload。 3'b000: 128 个字节 3'b001: 256 个字节 3'b010: 512 个字节 3'b011: 1024 个字节 3'b100: 2048 个字节 3'b101: 4096 个字节
3			保留

表 105: Config AXI User Max Payload Size (0x40) (续)

位索引	默认值	访问类型	描述
2:0	3'h5	RW	user_prg_payload 向用户应用发出的已编程最大有效载荷大小。 3'b000: 128 个字节 3'b001: 256 个字节 3'b010: 512 个字节 3'b011: 1024 个字节 3'b100: 2048 个字节 3'b101: 4096 个字节

### Config AXI User Max Read Request Size (0x44)

表 106: Config AXI User Max Read Request Size (0x44)

位索引	默认值	访问类型	描述
6:4	3'h5	RO	user_eff_read 向用户应用发出的最大读取请求大小。由于 PCIe 配置或数据路径宽度，该值可能低于 user_max_read。 3'b000: 128 个字节 3'b001: 256 个字节 3'b010: 512 个字节 3'b011: 1024 个字节 3'b100: 2048 个字节 3'b101: 4096 个字节
3			保留
2:0	3'h5	RW	user_prg_read 向用户应用发出的最大读取请求大小。 3'b000: 128 个字节 3'b001: 256 个字节 3'b010: 512 个字节 3'b011: 1024 个字节 3'b100: 2048 个字节 3'b101: 4096 个字节

### Config Write Flush Timeout (0x60)

表 107: Config Write Flush Timeout (0x60)

位索引	默认值	访问类型	描述
4:0	5'h0	RW	写入刷新超时 适用于 AXI4-Stream C2H 通道。此寄存器用于指定通道在刷新已从 PCIe 接收到的写入数据之前，等待数据的时钟周期数。此操作会关闭描述符并生成写回。值为 0 即禁用超时。时钟超时值 = $2^{\text{value}}$ 。

## H2C SGDMA 寄存器 (0x4)

表 108: H2C SGDMA 寄存器 (0x4)

地址 (十六进制)	寄存器名称
0x00	H2C SGDMA Identifier (0x00)
0x80	H2C SGDMA Descriptor Low Address (0x80)
0x84	H2C SGDMA Descriptor High Address (0x84)
0x88	H2C SGDMA Descriptor Adjacent (0x88)
0x8C	H2C SGDMA Descriptor Credits (0x8C)

## H2C SGDMA Identifier (0x00)

表 109: H2C SGDMA Identifier (0x00)

位索引	默认值	访问类型	描述
31:20	12'h1fc	RO	子系统标识符
19:16	4'h4	RO	H2C DMA 目标
15	1'b0	RO	数据流传输 1: AXI4-Stream 接口 0: AXI4 Memory Mapped 接口
14:12	3'h0	RO	保留
11:8	可变	RO	通道 ID 目标 [3:0]
7:0	8'h04	RO	版本 8'h01: 2015.3 和 2015.4 8'h02: 2016.1 8'h03: 2016.2 8'h04: 2016.3 8'h05: 2016.4 8'h06: 从 2017.1 到最新版本

## H2C SGDMA Descriptor Low Address (0x80)

表 110: H2C SGDMA Descriptor Low Address (0x80)

位索引	默认值	访问类型	描述
31:0	32'h0	RW	dsc_adr[31:0] 描述符起始地址的下半位元。Dsc_adr[63:0] 是在控制寄存器运行位已置位之后提取的首个描述符地址。

## H2C SGDMA Descriptor High Address (0x84)

表 111: H2C SGDMA Descriptor High Address (0x84)

位索引	默认值	访问类型	描述
31:0	32'h0	RW	dsc_adr[63:32] 描述符起始地址的上半位元。 Dsc_adr[63:0] 是在控制寄存器运行位已置位之后提取的首个描述符地址。

## H2C SGDMA Descriptor Adjacent (0x88)

表 112: H2C SGDMA Descriptor Adjacent (0x88)

位索引	默认值	访问类型	描述
5:0	6'h0	RW	dsc_adj[5:0] 表示位于描述符起始地址之后的额外相邻描述符数量。

## H2C SGDMA Descriptor Credits (0x8C)

表 113: H2C SGDMA Descriptor Credits (0x8C)

位索引	默认值	访问类型	描述
9:0	10'h0	RW	h2c_dsc_credit[9:0] 写入此寄存器将为通道添加描述符信用值。仅当在“描述符信用值模式”寄存器中通过通道的各个位启用此寄存器时，此寄存器才可供使用。 在通道控制寄存器运行位的下降沿上会自动清除信用值，或者针对通道禁用“描述符信用值模式”时，也同样如此。通过读取该寄存器即可判定该通道当前剩余信用值。

## C2H SGDMA 寄存器 (0x5)

本节描述了 C2H SGDMA 寄存器。

表 114: C2H SGDMA 寄存器 (0x5)

地址 (十六进制)	寄存器名称
0x00	C2H SGDMA Identifier (0x00)
0x80	C2H SGDMA Descriptor Low Address (0x80)
0x84	C2H SGDMA Descriptor High Address (0x84)
0x88	C2H SGDMA Descriptor Adjacent (0x88)
0x8C	C2H SGDMA Descriptor Credits (0x8C)

## C2H SGDMA Identifier (0x00)

表 115: C2H SGDMA Identifier (0x00)

位索引	默认值	访问类型	描述
31:20	12'h1fc	RO	子系统标识符
19:16	4'h5	RO	C2H DMA 目标
15	1'b0	RO	数据流传输 1: AXI4-Stream 接口 0: AXI4 Memory Mapped 接口
14:12	3'h0	RO	保留
11:8	可变	RO	通道 ID 目标 [3:0]
7:0	8'h04	RO	版本 8'h01: 2015.3 和 2015.4 8'h02: 2016.1 8'h03: 2016.2 8'h04: 2016.3 8'h05: 2016.4 8'h06: 从 2017.1 到最新版本

## C2H SGDMA Descriptor Low Address (0x80)

表 116: C2H SGDMA Descriptor Low Address (0x80)

位索引	默认值	访问类型	描述
31:0	32'h0	RW	dsc_adr[31:0] 描述符起始地址的下半位元。Dsc_adr[63:0] 是在控制寄存器运行位已置位之后提取的首个描述符地址。

## C2H SGDMA Descriptor High Address (0x84)

表 117: C2H SGDMA Descriptor High Address (0x84)

位索引	默认值	访问类型	描述
31:0	32'h0	RW	dsc_adr[63:32] 描述符起始地址的上半位元。 Dsc_adr[63:0] 是在控制寄存器运行位已置位之后提取的首个描述符地址。

## C2H SGDMA Descriptor Adjacent (0x88)

表 118: C2H SGDMA Descriptor Adjacent (0x88)

位索引	默认值	访问类型	描述
5:0	6'h0	RW	dsc_adj[5:0] 表示位于描述符起始地址之后的额外相邻描述符数量。



## C2H SGDMA Descriptor Credits (0x8C)

表 119: C2H SGDMA Descriptor Credits (0x8C)

位索引	默认值	访问类型	描述
9:0	10'h0	RW	c2h_dsc_credit[9:0] 写入此寄存器将为通道添加描述符信用值。仅当在“描述符信用值模式”寄存器中通过通道的各个位启用此寄存器时，此寄存器才可供使用。 在通道控制寄存器运行位的下降沿上会自动清除信用值，或者针对通道禁用“描述符信用值模式”时，也同样如此。通过读取该寄存器即可判定该通道当前剩余信用值。

## SGDMA 公用寄存器 (0x6)

表 120: SGDMA 公用寄存器 (0x6)

地址 (十六进制)	寄存器名称
0x00	SGDMA Identifier Registers (0x00)
0x10	SGDMA Descriptor Control Register (0x10)
0x14	SGDMA Descriptor Control Register (0x14)
0x18	SGDMA Descriptor Control Register (0x18)
0x20	SGDMA Descriptor Credit Mode Enable (0x20)
0x24	SG Descriptor Mode Enable Register (0x24)
0x28	SG Descriptor Mode Enable Register (0x28)

## SGDMA Identifier Registers (0x00)

表 121: SGDMA Identifier Registers (0x00)

位索引	默认值	访问类型	描述
31:20	12'h1fc	RO	子系统标识符
19:16	4'h6	RO	SGDMA 目标
15:8	8'h0	RO	保留
7:0	8'h04	RO	版本 8'h01: 2015.3 和 2015.4 8'h02: 2016.1 8'h03: 2016.2 8'h04: 2016.3 8'h05: 2016.4 8'h06: 从 2017.1 到最新版本

## SGDMA Descriptor Control Register (0x10)

表 122: SGDMA Descriptor Control Register (0x10)

位索引	默认值	访问类型	描述
19:16	4'h0	RW	c2h_dsc_halt[3:0] 每条 C2H 通道 1 位。设为 1 即可为对应通道停止描述符提取。
15:4			保留
3:0	4'h0	RW	h2c_dsc_halt[3:0] 每条 H2C 通道 1 位。设为 1 即可为对应通道停止描述符提取。

## SGDMA Descriptor Control Register (0x14)

表 123: SGDMA Descriptor Control Register (0x14)

位索引	默认值	访问类型	描述
		W1S	位元描述与 SGDMA Descriptor Control Register (0x10) 中的位元描述相同。

## SGDMA Descriptor Control Register (0x18)

表 124: SGDMA Descriptor Control Register (0x18)

位索引	默认值	访问类型	描述
		W1C	位元描述与 SGDMA Descriptor Control Register (0x10) 中的位元描述相同。

## SGDMA Descriptor Credit Mode Enable (0x20)

表 125: SGDMA Descriptor Credit Mode Enable (0x20)

位索引	默认值	访问类型	描述
3:0	0x0	RW	h2c_dsc_credit_enable [3:0] 每条 H2C 通道 1 位。设为 1 即可启用描述符信用值计算。对于每条通道，描述符提取引擎将把提取的描述符数量限制为通过写入该通道的“描述符信用值寄存器”赋予该描述符的信用值数量。
15:4			保留
19:16	0x0	RW	c2h_dsc_credit_enable [3:0] 每条 C2H 通道 1 位。设为 1 即可启用描述符信用值计算。对于每条通道，描述符提取引擎将把提取的描述符数量限制为通过写入该通道的“描述符信用值寄存器”赋予该描述符的信用值数量。

## SG Descriptor Mode Enable Register (0x24)

表 126: SG Descriptor Mode Enable Register (0x24)

位索引	默认值	访问类型	描述
		W1S	位元描述与 SGDMA Descriptor Credit Mode Enable (0x20) 中的位元描述相同。

## SG Descriptor Mode Enable Register (0x28)

表 127: SG Descriptor Mode Enable Register (0x28)

位索引	默认值	访问类型	描述
		W1C	位元描述与 SGDMA Descriptor Credit Mode Enable (0x20) 中的位元描述相同。

## MSI-X 矢量表和 PBA (0x8)

MSI-X 矢量表核 PBA 如下表所述。MSI-X 表偏移起始位置为 0x8000。下表显示了 2 个 MSI-X 矢量条目（MSI-X 表包含 32 个矢量条目）。PBA 地址偏移起始位置为 0x8FE0。地址偏移为固定值。

**注释：**配置控制寄存器中的 MSI-X 使能应先断言有效，然后再写入 MSI-X 表。否则，MSI-X 表将无法按期望方式运行。

表 128: MSI-X 矢量表和 PBA (0x00–0xFE0)

字节偏移	位索引	默认值	访问类型	描述
0x00	31:0	32'h0	RW	MSIX_Vector0_Address[31:0] MSI-X vector0 消息下位地址。
0x04	31:0	32'h0	RW	MSIX_Vector0_Address[63:32] MSI-X vector0 消息上位地址。
0x08	31:0	32'h0	RW	MSIX_Vector0_Data[31:0] MSI-X vector0 消息数据。
0x0C	31:0	32'hFFFFFFFF	RW	MSIX_Vector0_Control[31:0] MSI-X vector0 控制。 位元位置： 31:1：保留。 0：掩码。设为 1 时，此 MSI-X 矢量不用于生成消息。 复位为 0 时，此 MSI-X 矢量用于生成消息。
0x1F0	31:0	32'h0	RW	MSIX_Vector31_Address[31:0] MSI-X vector31 消息下位地址。
0x1F4	31:0	32'h0	RW	MSIX_Vector31_Address[63:32] MSI-X vector31 消息上位地址。
0x1F8	31:0	32'h0	RW	MSIX_Vector31_Data[31:0] MSI-X vector31 消息数据。

表 128：MSI-X 矢量表和 PBA (0x00–0xFE0) (续)

字节偏移	位索引	默认值	访问类型	描述
0x1FC	31:0	32'hFFFFFFFF	RW	MSIX_Vector31_Control[31:0] MSI-X vector31 控制。 位元位置： 31:1：保留。 0：掩码。置位为 1 时，此 MSI-X 矢量不用于生成消息。复位为 0 时，此 MSI-X 矢量用于生成消息。
0xFE0	31:0	32'h0	RW	Pending_Bit_Array[31:0] MSI-X 暂挂位阵列。每个矢量均有 1 个位。位 0 对应于 vector0，以此类推。

# 利用子系统进行设计

本节包含有关利用该子系统来协助简化设计的指导信息和其它信息。

## 时钟设置和复位

### 时钟设置

`axi_aclk` 输出为用于所有 AXI 接口的时钟，应驱动所有对应的 AXI Interconnect `aclk` 信号。`axi_aclk` 并非自由运行的时钟。这是衍生时钟，当 `axi_aresetn` 信号断言无效后，此时钟有效。

**注释：**`axi_aclk` 输出不应用作为您的设计的系统时钟。`axi_aclk` 并非自由运行的时钟输出。如上所述，`axi_aclk` 并非始终存在。

### 复位

对于 AXI Bridge 模式下的 DMA/ Bridge Subsystem for PCIe，存在可选 `dma_bridge_resetn` 输入管脚，此管脚支持您复位所有内部 Bridge 引擎和寄存器以及由 `axi_aresetn` 管脚驱动的所有 AXI 外设。设置以下参数后，在初始链路正常运行期间，`dma_bridge_resetn` 无需断言有效，因为此操作将由 IP 自动完成。您必须先终止所有传输事务，然后才能断言此管脚有效。此管脚断言有效后必须在一段时间内保持处于断言有效状态，此持续时间至少等于“完成超时 (Completion Timeout)”值（通常为 50 ms），这样才能将当前数据路径中可能处于排队状态的所有暂挂传输清除。要设置此参数，请在 Tcl 命令行中输入以下命令：

```
set_property -dict [list CONFIG.soft_reset_en {true}] [get_ips <ip_name>]
```

如需了解有关时钟设置和复位的信息，请参阅相应的 PCIe® 集成块产品指南：

- 《7 系列 FPGA Integrated Block for PCI Express LogiCORE IP 产品指南》(PG054)
- 《Virtex-7 FPGA Integrated Block for PCI Express LogiCORE IP 产品指南》(PG023)
- 《UltraScale 器件 Gen3 Integrated Block for PCI Express LogiCORE IP 产品指南》(PG156)
- 《UltraScale+ Integrated Block for PCI Express LogiCORE IP 产品指南》(PG213)

## 串联配置

串联配置 (Tandem Configuration) 功能可供赛灵思 DMA Subsystem for PCI Express® 用于所有 UltraScale™ 器件和大部分 UltraScale+™ 器件。串联配置使用 2 阶方法以支持 IP 满足 PCI Express 规范中所述的配置时间要求。此技术支持多种用例，如下所述：

- 串联 PROM (Tandem PROM)：从闪存加载单一 2 阶比特流。

- 串联 PCIe (Tandem PCIe)：从闪存加载第一阶段比特流，并通过 PCIe 链路将第二阶段比特流交付至 MCAP。
- 含现场更新的串联 (Tandem with Field Updates)：完成串联 PROM（仅限 UltraScale）或串联 PCIe 初始配置后，在 PCIe 链路保持有效时更新整个用户设计。更新区域（布局图）和设计结构均已预定义，并且已提供 Tcl 脚本。
- 串联 + Dynamic Function eXchange：这是较常用的用例，针对任意大小或任意数量的动态区域采用串联配置后接 Dynamic Function eXchange (DFX)。
- 基于 PCIe 的 Dynamic Function eXchange：这是标准配置，后接 DFX，使用 PCIe/MCAP 作为部分比特流的交付路径。

如需了解有关 Dynamic Function eXchange 的信息，请参阅《Vivado Design Suite 用户指南：Dynamic Function eXchange》(UG909)。

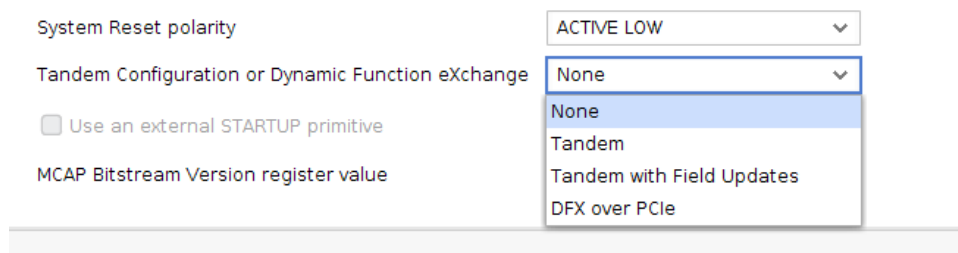
## 自定义子系统以启用串联配置

### UltraScale 器件

要为 UltraScale™ 器件启用任意“串联配置 (Tandem Configuration)”功能，请在自定义该子系统时选择相应的 Vivado® IP 目录选项。在“基本信息 (Basic)”选项卡中：

1. 将“模式 (Mode)”切换为“Advanced”。
2. 根据特定用例更改 Tandem Configuration 或 Dynamic Function eXchange 选项：
  - 串联 (Tandem)：适用于 Tandem PROM、Tandem PCIe 或 Tandem + Dynamic Function eXchange 用例。
  - 含现场更新的串联 (Tandem with Field Updates)：仅适用于预定义的 Field Updates 用例。
  - 基于 PCIe 的 DFX (DFX over PCIe)：用于为 Dynamic Function eXchange 启用 MCAP 链路，而无需启用 Tandem Configuration。

图 10：适用于 UltraScale 器件的串联配置或 Dynamic Function eXchange 选项



如需了解有关 Tandem Configuration 的完整信息（包括所需的 PCIe 块位置、设计流程示例、要求、限制和其它考量因素），请参阅《UltraScale 器件 Gen3 Integrated Block for PCI Express LogiCORE IP 产品指南》(PG156) 中的 [串联配置](#)。

### UltraScale+ 器件

要为 UltraScale+™ 器件启用任意“串联配置 (Tandem Configuration)”功能，请在自定义该子系统时选择相应的 IP 目录选项。在“基本信息 (Basic)”选项卡中：

1. 将“模式 (Mode)”切换为“Advanced”。
2. 根据特定用例更改 Tandem Configuration 或 Dynamic Function eXchange 选项：
  - 串联 PROM (Tandem PROM)：适用于 Tandem PROM 用例。

- 串联 PCIe (Tandem PCIe)：适用于 Tandem PCIe 或 Tandem + Dynamic Function eXchange 用例。
- 含现场更新的串联 PCIe：仅适用于预定义的 Field Updates 用例。
- 基于 PCIe 的 DFX (DFX over PCIe)：用于为 Dynamic Function eXchange 启用 MCAP 链路，而无需启用 Tandem Configuration。

图 11：Tandem Configuration 或 Dynamic Function eXchange 选项

System Reset polarity

ACTIVE LOW

Tandem Configuration or Dynamic Function eXchange

Tandem PCIe

None

Tandem PROM

Tandem PCIe

Tandem PCIe with Field Updates

DFX over PCIe

MCAP Bitstream Version register value



**重要提示！** Tandem Configuration 当前受 DMA 模式支持，但在 UltraScale+ 器件中不受 Bridge 模式支持。

如需了解有关 Tandem Configuration 的完整信息（包括所需的 PCIe 块位置、设计流程示例、要求、限制和其它考量因素），请参阅《UltraScale+ Integrated Block for PCI Express LogiCORE IP 产品指南》(PG213) 中的[串联配置](#)。

## 支持的器件

DMA/Bridge Subsystem for PCIe® 和 Vivado® 工具流程支持以赛灵思参考板和特定器件/封装组合为目标的实现。串联配置支持下列各表中所示配置。

### UltraScale 器件

下表列出了 UltraScale™ 器件支持的串联 PROM/PCIe 配置。

<b>HDL</b>	仅限 Verilog				
<b>PCIe 配置</b>	所有配置（最大：X8Gen3）				
<b>赛灵思参考开发板支持</b>	适用于 Kintex® UltraScale™ FPGA 的 KCU105 评估板 适用于 Virtex® UltraScale™ FPGA 的 VCU108 评估板				
<b>器件支持</b>	<b>器件<sup>1</sup></b>	<b>PCIe 块位置</b>	<b>PCIe 复位位置</b>	<b>串联配置</b>	<b>含现场更新的串联</b>
Kintex UltraScale	XCKU025	PCIE_3_1_X0Y0	IOB_X1Y103	量产	量产
	XCKU035	PCIE_3_1_X0Y0	IOB_X1Y103	量产	量产
	XCKU040	PCIE_3_1_X0Y0	IOB_X1Y103	量产	量产
	XCKU060	PCIE_3_1_X0Y0	IOB_X2Y103	量产	量产
	XCKU085	PCIE_3_1_X0Y0	IOB_X2Y103	量产	量产
	XCKU095	PCIE_3_1_X0Y0	IOB_X1Y103	量产	量产
	XCKU115	PCIE_3_1_X0Y0	IOB_X2Y103	量产	量产

Virtex UltraScale	XCVU065	PCIE_3_1_X0Y0	IOB_X1Y103	量产	量产
	XCVU080	PCIE_3_1_X0Y0	IOB_X1Y103	量产	量产
	XCVU095	PCIE_3_1_X0Y0	IOB_X1Y103	量产	量产
	XCVU125	PCIE_3_1_X0Y0	IOB_X1Y103	量产	量产
	XCVU160	PCIE_3_1_X0Y1	IOB_X1Y363	量产	量产
	XCVU190	PCIE_3_1_X0Y2	IOB_X1Y363	量产	量产
	XCVU440	PCIE_3_1_X0Y2	IOB_X1Y363	量产	量产

**注释：**

1. 仅针对量产硅片提供正式支持。针对所有工程样品硅片 (ES2) 器件禁用比特流生成。

## UltraScale+ 器件

下表列出了 UltraScale+ 器件支持的串联 PROM/PCIe 配置。

<b>HDL</b>	仅限 Verilog			
<b>PCIe 配置</b>	所有配置 (最大: X16Gen3 或 X8Gen4)			
<b>赛灵思参考开发板支持</b>	适用于 Kintex UltraScale+ FPGA 的 KCU116 评估板 适用于 Virtex UltraScale+ FPGA 的 VCU118 评估板			
<b>器件支持</b>	<b>器件 <sup>1</sup></b>	<b>PCIe 块位置</b>	<b>串联配置</b>	<b>含现场更新的串联 PCIe</b>
Kintex UltraScale+	KU3P	PCIE40E4_X0Y0	量产	量产
	KU5P	PCIE40E4_X0Y0	量产	量产
	KU11P	PCIE40E4_X1Y0	量产	量产
	KU15P	PCIE40E4_X1Y0	量产	量产
	KU19P <sup>2</sup>	不适用	不支持	不支持
Virtex UltraScale+	VU3P	PCIE40E4_X1Y0	量产	量产
	VU5P	PCIE40E4_X1Y0	量产	量产
	VU7P	PCIE40E4_X1Y0	量产	量产
	VU9P	PCIE40E4_X1Y2	量产	量产
	VU11P	PCIE40E4_X0Y0	量产	量产
	VU13P	PCIE40E4_X0Y1	量产	量产
	VU19P	PCIE4CE4_X0Y2	暂不支持	暂不支持
	VU23P	PCIE40E4_X0Y0	量产	量产
	VU27P	PCIE40E4_X0Y0	量产	量产
	VU29P	PCIE40E4_X0Y0	量产	量产
	VU31P	PCIE4CE4_X1Y0	量产	量产
	VU33P	PCIE4CE4_X1Y0	量产	量产
	VU35P	PCIE4CE4_X1Y0	量产	量产
	VU37P	PCIE4CE4_X1Y0	量产	量产
	VU45P	PCIE4CE4_X1Y0	量产	量产
	VU47P	PCIE4CE4_X1Y0	量产	量产
	VU57P	PCIE4CE4_X1Y0	量产	量产



Zynq® UltraScale+™ MPSoC	ZU4CG/EG/EV	PCIE40E4_X0Y1	量产	量产
	ZU5CG/EG/EV	PCIE40E4_X0Y1	量产	量产
	ZU7CG/EG/EV	PCIE40E4_X0Y1	量产	量产
	ZU11EG	PCIE40E4_X1Y0	量产	量产
	ZU17EG	PCIE40E4_X1Y0	量产	量产
	ZU19EG	PCIE40E4_X1Y0	量产	量产
Zynq® UltraScale+™ RFSoc <sup>2</sup>	ZU21DR	PCIE40E4_X0Y0	不支持	不支持
	ZU25DR	PCIE40E4_X0Y0	不支持	不支持
	ZU27DR	PCIE40E4_X0Y0	不支持	不支持
	ZU28DR	PCIE40E4_X0Y0	不支持	不支持
	ZU29DR	PCIE40E4_X0Y0	不支持	不支持
	ZU39DR	PCIE4CE4_X0Y0	不支持	不支持
	ZU43DR	PCIE4CE4_X0Y0	不支持	不支持
	ZU45DR	PCIE4CE4_X0Y0	不支持	不支持
	ZU47DR	PCIE4CE4_X0Y0	不支持	不支持
	ZU48DR	PCIE4CE4_X0Y0	不支持	不支持
	ZU49DR	PCIE4CE4_X0Y0	不支持	不支持

**注释：**

1. 仅针对量产硅片提供正式支持。针对所有工程样品硅片（ES1 和 ES2）器件禁用比特流生成。
2. Kintex UltraScale+ KU19P 和所有 Zynq RFSoc 器件都不含启用 MCAP 的 PCIe 块位置。有鉴于此，当前这些器件不支持串联配置 (Tandem Configuration)。

## 设计流程步骤

本章节描述了子系统的自定义和生成方式、子系统的约束方式以及此 IP 子系统的仿真、综合与实现的具体步骤。如需获取有关标准 Vivado® 设计流程以及有关 IP integrator 的详细信息，请参阅以下 Vivado Design Suite 用户指南：

- 《Vivado Design Suite 用户指南：采用 IP integrator 设计 IP 子系统》(UG994)
- 《Vivado Design Suite 用户指南：采用 IP 进行设计》(UG896)
- 《Vivado Design Suite 用户指南：入门指南》(UG910)
- 《Vivado Design Suite 用户指南：逻辑仿真》(UG900)

---

## 自定义和生成子系统

本节包含有关如何使用赛灵思工具在 Vivado® Design Suite 中自定义和生成子系统的信息。

如果要在 Vivado IP integrator 中自定义和生成该子系统，请参阅《Vivado Design Suite 用户指南：采用 IP integrator 设计 IP 子系统》(UG994)以获取详细信息。验证或生成设计时，IP integrator 可能会自动计算某些配置值。要查看配置值是否更改，请参阅本章中的参数说明。要查看参数值，请在 Tcl 控制台 (Tcl console) 中运行 `validate_bd_design` 命令。

您可以遵循以下步骤通过指定与 IP 子系统关联的各种参数值来自定义设计中使用的 IP：

1. 从 IP 目录 (IP catalog) 中选择 IP。
2. 双击所选 IP、从工具栏或右键单击菜单中选择“自定义 IP (Customize IP)”命令。

欲知详情，请参阅《Vivado Design Suite 用户指南：采用 IP 进行设计》(UG896) 和《Vivado Design Suite 用户指南：入门指南》(UG910)。

本章中的附图是 Vivado IDE 的插图。此处展示的布局可能与当前版本中的布局有所不同。

### “Basic” 选项卡

DMA 模式的“基本信息 (Basic)”选项卡（“功能模式 (Functional Mode)”选项）如下图所示。

图 12：DMA 功能模式的“Basic”选项卡

Component Name: xdma\_0

**Basic** | PCIe ID | PCIe : BARs | PCIe : MISC | PCIe : DMA | Debug Options | Shared Logic | GT Settings

Functional Mode: DMA

Mode: Advanced

Device / Port Type: PCI Express Endpoint device

PCIe Block Location: X1Y2

**GT Selection**

☐ Enable GT Quad Selection

GT Quad: GTY Quad 227

**PCIe Interface**

Lane Width: X8

**Maximum Link Speed**

☐ 2.5 GT/s ☐ 5.0 GT/s ☒ 8.0 GT/s

Reference Clock Frequency (MHz): 100 MHz

Reset Source: User Reset

GT DRP Clock Selection: Internal

**AXI Interface**

AXI Address Width: 64 [32 - 64]

AXI Data Width: 256 bit

AXI Clock Frequency: 250

**DMA Interface option**

☒ AXI Memory Mapped ☐ AXI Stream

☐ AXI-Lite Slave Interface

**Data Protection**

☒ None ☐ Check Parity ☐ Propagate Parity

☒ Enable PIPE Simulation

☐ Enable GT Channel DRP Ports

☐ Enable PCIe DRP Ports

☐ Additional Transceiver Control and Status Ports

System Reset polarity: ACTIVE LOW

Tandem Configuration or Dynamic Function eXchange: None

MCAP Bitstream Version register value: 00000000

其选项具体定义如下：

- 功能模式 (Functional Mode)：允许您选择：
  - DMA (DMA Subsystem for PCIe)。

- AXI Bridge (AXI Bridge Subsystem for PCIe)。“AXI Bridge”选项仅对 UltraScale+™ 器件有效。如需了解有关 PCIe Bridge 模式操作的详细信息，请参阅《AXI Bridge for PCI Express Gen3 Subsystem 产品指南》(PG194)。本文档仅涵盖 DMA 模式操作。
- 模式 (Mode)：允许您为子系统的配置选择“基本 (Basic)”模式或“高级 (Advanced)”模式。
- 器件/端口类型 (Device /Port Type)：仅支持“PCI Express® Endpoint device”模式。
- PCIe 块位置 (PCIe Block Location)：从可用集成块中进行选择，以支持生成特定位置的约束文件和管脚分配。该选项可在默认设计示例脚本中使用。如果选中“赛灵思开发板 (Xilinx Development Board)”，则该选项不可用。
- 通道宽度 (Lane Width)：子系统需要选择初始通道宽度。要了解受支持的通道宽度和链路速度，请参阅《7 系列 FPGA Integrated Block for PCI Express LogiCORE IP 产品指南》(PG054)、《Virtex-7 FPGA Integrated Block for PCI Express LogiCORE IP 产品指南》(PG023)、《UltraScale 器件 Gen3 Integrated Block for PCI Express LogiCORE IP 产品指南》(PG156) 或《UltraScale+ Integrated Block for PCI Express LogiCORE IP 产品指南》(PG213)。核设置的链路速度越快，那么连接到支持更低链路速度的器件时，就可以训练至更低的链路速度。
- 最大链路速度 (Maximum Link Speed)：子系统需要选择 PCIe Gen 速度。
- 参考时钟频率 (Reference Clock Frequency)：默认值为 100 MHz，但也支持 125 MHz 和 250 MHz。
- 复位源 (Reset Source)：可选择“用户复位 (User Reset)”或“物理就绪 (Phy ready)”。
- 建立链路后，用户复位来自于 PCIe 核。当 PCIe 链路中断时，“User Reset”会断言有效，且 XDMA 会转为复位模式。当链路恢复时，“User Reset”会断言无效。
- 选中“Phy ready”选项时，XDMA 不受 PCIe 链路状态影响。
- GT DRP 时钟选择 (GT DRP Clock Selection)：选择内部时钟（默认选项）或外部时钟。
- GT 选择 (GT Selection)：启用 GT 四通道选择 (Enable GT Quad Selection)：选择通道 0 所在的四通道。
- AXI 地址宽度 (AXI Address Width)：当前，仅支持 64 位宽度。
- AXI 数据宽度 (AXI Data Width)：选择 64、128、256 位或 512 位（仅适用于 UltraScale+）。该子系统允许您选择“接口宽度 (Interface Width)”，如需了解其定义，请参阅《7 系列 FPGA Integrated Block for PCI Express LogiCORE IP 产品指南》(PG054)、《Virtex-7 FPGA Integrated Block for PCI Express LogiCORE IP 产品指南》(PG023)、《UltraScale 器件 Gen3 Integrated Block for PCI Express LogiCORE IP 产品指南》(PG156) 或《UltraScale+ Integrated Block for PCI Express LogiCORE IP 产品指南》(PG213)。
- AXI 时钟频率 (AXI Clock Frequency)：根据通道宽度/速度，可选择 62.5 MHz、125 MHz 或 250 MHz。
- DMA 接口选项 (DMA Interface Option)：选择 AXI4 Memory Mapped 或 AXI4-Stream。
- AXI4-Lite 从接口 (AXI4-Lite Slave Interface)：选择此项即可启用 AXI4-Lite 从接口。
- 数据保护 (Data Protection)：默认禁用奇偶校验检查。
- 启用“Check Parity”时，XDMA 在读取来自 PCIe 的数据时执行奇偶校验检查，并在将数据写入 PCIe 时生成奇偶校验。
- 启用“Propagate Parity”时，XDMA 会将奇偶校验传输至用户 AXI 接口。用户负责在用户 AXI 接口上检查并生成奇偶校验。
- 串联配置 (Tandem Configuration) 或 Dynamic Function eXchange：选择串联配置或 Dynamic Function eXchange 功能（如果适用于您的设计）。

## 相关信息

## 串联配置

## “PCIe ID” 选项卡

“PCIe ID” 选项卡如下图所示。

图 13: “PCIe ID” 选项卡

如需获取这些选项的描述，请参阅下列相应产品指南中的“设计流程步骤”章节：

- 《7 系列 FPGA Integrated Block for PCI Express LogiCORE IP 产品指南》(PG054)
- 《Virtex-7 FPGA Integrated Block for PCI Express LogiCORE IP 产品指南》(PG023)
- 《UltraScale 器件 Gen3 Integrated Block for PCI Express LogiCORE IP 产品指南》(PG156)
- 《UltraScale+ Integrated Block for PCI Express LogiCORE IP 产品指南》(PG213)

## “PCIe BARs” 选项卡

“PCIe BARs” 选项卡如下图所示。

图 14: “PCIe BARs” 选项卡

Component Name: xdma\_0

Board Basic PCIe ID **PCIe : BARs** PCIe : MISC PCIe : DMA

☒ PCIe to AXI Lite Master Interface

☐ 64bit Enable ☐ Prefetchable

Size: 1 Scale: Megabytes

Value: FFF00000

PCIe to AXI Translation: 0x0000000000000000

☒ PCIe to DMA Interface

☐ 64bit Enable ☐ Prefetchable

☒ PCIe to DMA Bypass Interface

☐ 64bit Enable ☐ Prefetchable

Size: 1 Scale: Megabytes

Value: FFF00000

PCIe to AXI Translation: 0x0000000000000000

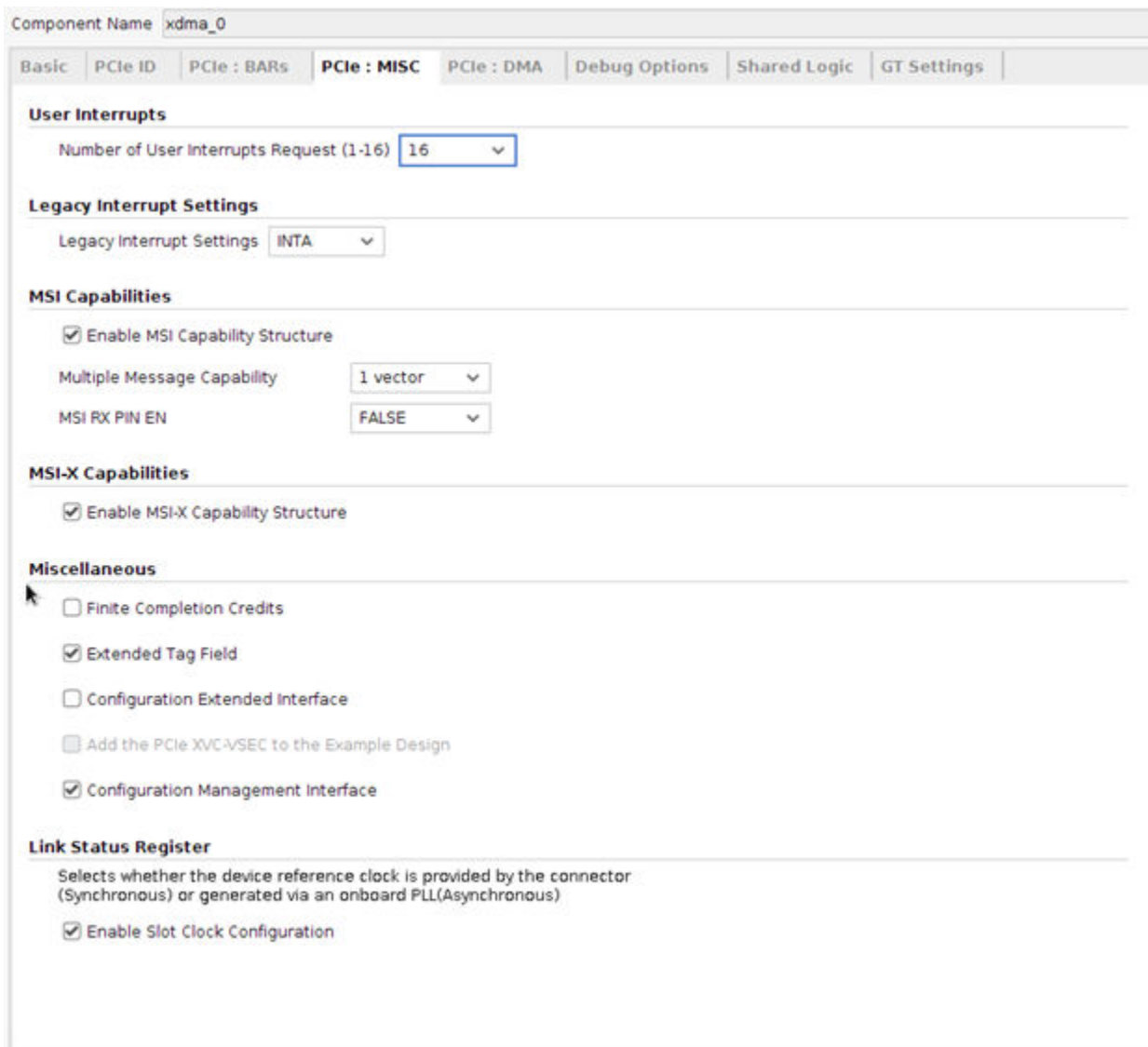
- PCIe to AXI Lite Master Interface: (可选) 您可启用该“PCIe 到 AXI Lite 主接口”选项以连接至 AXI-Lite 接口 BAR 空间。大小 (size)、单位 (scale) 和地址转换 (address translation) 均可配置。
- PCIe to XDMA Interface: 该“PCIe 到 XDMA 接口”选项始终选中。
- PCIe to DMA Bypass Interface: (可选) 您可启用“PCIe 到 DMA 旁路接口”BAR 空间。大小、单位和地址转换同样均可配置。

可单独选中每个 BAR 空间以设置 64 位选项。每个 64 位 BAR 空间均可选择是否可预取 (Prefetchable)。

## “PCIe MISC” 选项卡

“PCIe Miscellaneous” 选项卡如下图所示。

图 15: “PCIe Misc” 选项卡



Component Name: xdma\_0

Basic | PCIe ID | PCIe : BARs | **PCIe : MISC** | PCIe : DMA | Debug Options | Shared Logic | GT Settings

**User Interrupts**

Number of User Interrupts Request (1-16): 16

**Legacy Interrupt Settings**

Legacy Interrupt Settings: INTA

**MSI Capabilities**

☒ Enable MSI Capability Structure

Multiple Message Capability: 1 vector

MSI RX PIN EN: FALSE

**MSI-X Capabilities**

☒ Enable MSI-X Capability Structure

**Miscellaneous**

☐ Finite Completion Credits

☒ Extended Tag Field

☐ Configuration Extended Interface

☐ Add the PCIe XVC/VSEC to the Example Design

☒ Configuration Management Interface

**Link Status Register**

Selects whether the device reference clock is provided by the connector (Synchronous) or generated via an onboard PLL (Asynchronous)

☒ Enable Slot Clock Configuration

- 用户中断请求数 (Number of User Interrupt Request): 可选用户中断请求最大数量为 16。
- 传统中断设置 (Legacy Interrupt Settings): 选择任一传统中断: INTA、INTB、INTC 或 INTD。
- MSI 功能 (MSI Capabilities): 默认启用 “MSI Capabilities”，并启用 1 个矢量。可选矢量最大数量为 32。一般情况下，Linux 针对 MSI 仅使用 1 个矢量。该选项可禁用。
- MSI RX PIN EN: 该选项仅在 AXI Bridge Root Port 模式下有效。
- MSI-X 功能 (MSI-X Capabilities): 选择 MSI-X 事件。如需了解更多信息，请参阅 [MSI-X 矢量表](#) 和 [PBA \(0x8\)](#)。
- 有限完成信用值 (Finite Completion Credits): 在支持有限完成信用值的系统上，可启用该选项以改善性能。
- 扩展标签字段 (Extended Tag Field): 默认情况下使用 6 位完成标签。对于 UltraScale™ 和 Virtex®-7 器件，“扩展标签 (Extended Tag)” 选项可提供 64 个标签。对于 UltraScale+™ 器件，“扩展标签 (Extended Tag)” 选项可提供 256 个标签。如果不选中 “Extended Tag”，DMA 会针对所有器件使用 32 个标签。

- 配置扩展接口 (Configuration Extend Interface)：可选择 PCIe 扩展接口以增加配置空间。选中 “Configuration Extend Interface” 时，由您负责添加接口扩展逻辑以使其正常工作。
- 配置管理接口 (Configuration Management Interface)：选中该选项时，即可将 PCIe 配置管理接口置于顶层。
- 链路状态寄存器 (Link Status Register)：默认情况下，选中 “Enable Slot Clock Configuration”。这意味着在链路状态寄存器中启用插槽配置位。

## “PCIe DMA” 选项卡

“PCIe DMA” 选项卡如下图所示。

图 16：“PCIe DMA” 选项卡

Component Name: xdma\_0

Basic	PCIe ID	PCIe : BARs	PCIe : MISC	PCIe : DMA	Debug Options	Shared Logic	GT Settings
				Number of DMA Read Channel (H2C)	4		
				Number of DMA Write Channel (C2H)	4		
				Number of Request IDs for Read channel (2,4,8,16,32,64)	32	[2 - 64]	
				Number of Request IDs for Write channel (2,4,8,16,32)	16	[2 - 32]	
				Descriptor Bypass for Read (H2C)	0000		
				Descriptor Bypass for Write (C2H)	0000		
				AXI ID Width	4		
				<input type="checkbox"/> DMA Status Ports			

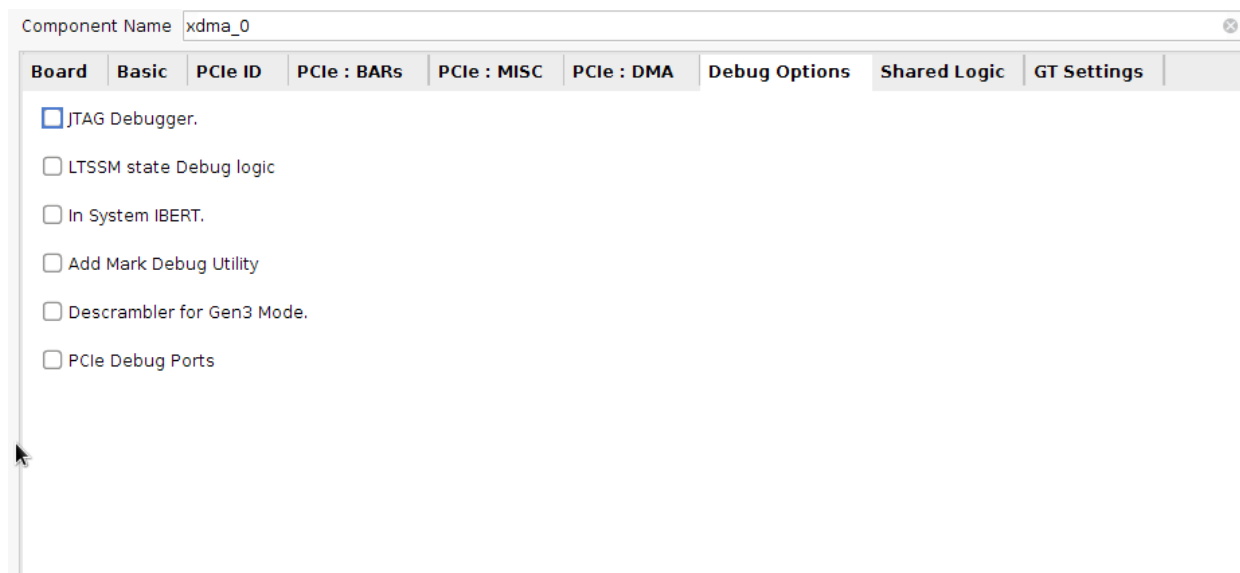
- DMA 读取通道数 (Number of DMA Read Channels)：可用选项范围为 1 到 4。
- DMA 写入通道数 (Number of DMA Write Channels)：可用选项范围为 1 到 4。
- 读取通道的请求 ID 数 (Number of Request IDs for Read channel)：选择每个通道未完成的请求的最大数量。可用选项范围为 2 到 64。
- 写入通道的请求 ID 数 (Number of Request IDs for Write channel)：选择每个通道未完成的请求的最大数量。可用选项范围为 2 到 32。
- 对应读取的描述符旁路 (Descriptor Bypass for Read) (H2C)：可用于所有选定的读取通道。每个二进制数字均对应于 1 条通道。LSB 对应于通道 0。位元位置的值为 1 表示对应的通道已启用描述符旁路。
- 对应写入的描述符旁路 (Descriptor Bypass for Write) (C2H)：可用于所有选定的写入通道。每个二进制数字均对应于 1 条通道。LSB 对应于通道 0。位元位置的值为 1 表示对应的通道已启用描述符旁路。
- AXI ID 宽度 (AXI ID Width)：默认位宽为 4 位。您也可以选择 2 位。
- DMA 状态端口 (DMA Status port)：DMA 状态端口可用于所有通道。



## “Debug Options” 选项卡

“调试选项 (Debug Options)” 选项卡如下图所示。

图 17: “Debug Options” 选项卡



- JTAG 调试器：该选项用于启用 JTAG 调试。
- LTSSM State Debug Logic：该选项可显示从链路建立开始后执行的所有 LTSSM 状态转换。
- In-System IBERT：该选项用于检查串行链路的眼图，确认该链路是否按期望的链路速度运行。如需了解有关 In-System IBERT 的更多信息，请参阅《In-System IBERT LogiCORE IP 产品指南》(PG246)。



**重要提示！** 该选项主要用于硬件调试。使用该选项时，不支持仿真。

- Add Mark Debug Utility：该选项可通过 mark\_debug 属性添加预定义的 PCIe 信号，以便在 ILA 中添加这些信号用于调试。
- Descrambler for Gen3 Mode：该选项在 PCIe 核内集成解扰器模块的加密版本，用于对采用 Gen3 链路速度模式往来 PCIe 集成块的 PIPE 数据进行解扰。

- PCIe Debug Ports：启用该选项后，下列端口即可用：

- cfg\_negotiated\_width: cfg\_negotiated\_width\_o
- cfg\_current\_speed: cfg\_current\_speed\_o
- cfg\_ltssm\_state: cfg\_ltssm\_state\_o
- cfg\_err\_cor: cfg\_err\_cor\_o
- cfg\_err\_fatal: cfg\_err\_fatal\_o
- cfg\_err\_nonfatal: cfg\_err\_nonfatal\_o
- cfg\_local\_error: cfg\_local\_error\_o
- cfg\_local\_error\_valid: cfg\_local\_error\_valid\_o

## “Shared Logic” 选项卡

下图显示了 UltraScale™ 器件中对应 IP 的“共享逻辑 (Shared Logic)”选项卡。

图 18：共享逻辑 (UltraScale 器件)

The screenshot shows the 'Shared Logic' configuration window for the 'xdma\_0' component. The window has a tabbed interface with tabs for 'Basic', 'PCIe ID', 'PCIe : BARs', 'PCIe : MISC', 'PCIe : DMA', 'Debug Options', 'Shared Logic' (selected), and 'GT Settings'. Under the 'Shared Logic' tab, there are two sections: 'Shared Logic GT\_COMMON' and 'GT Wizard Option'. In the 'Shared Logic GT\_COMMON' section, the radio button 'Include Shared Logic in core' is selected. In the 'GT Wizard Option' section, the radio button 'Include GT Wizard in core' is selected.

下图显示了 UltraScale+™ 器件中对应 IP 的“共享逻辑 (Shared Logic)”选项卡。

图 19：共享逻辑 (UltraScale+ 器件)

The screenshot shows the 'Shared Logic' configuration window for the 'xdma\_0' component. The window has a tabbed interface with tabs for 'Basic', 'PCIe ID', 'PCIe : BARs', 'PCIe : MISC', 'PCIe : DMA', 'Debug Options', 'Shared Logic' (selected), and 'GT Settings'. Under the 'Shared Logic' tab, there are two sections: 'GT Wizard Option' and 'GT COMMON Option'. In the 'GT Wizard Option' section, the radio button 'Include GT Wizard in core' is selected. In the 'GT COMMON Option' section, the radio button 'No sharing when inside GT Wizard and PCIe' is selected.

如需获取这些选项的描述，请参阅下列相应产品指南中的第 4 章节“设计流程步骤”：

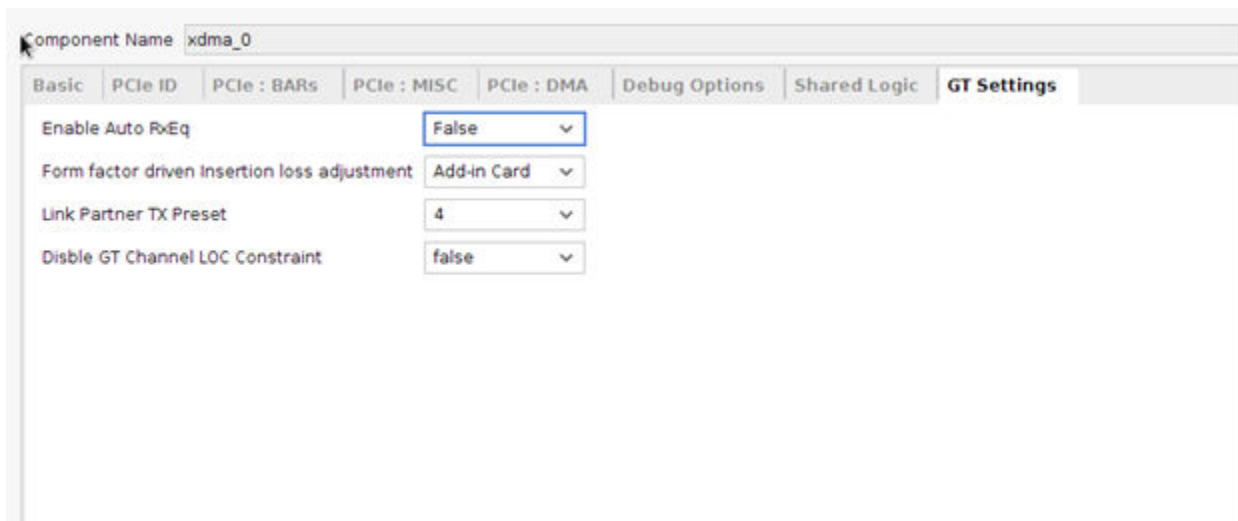
- 《UltraScale 器件 Gen3 Integrated Block for PCI Express LogiCORE IP 产品指南》(PG156)

- 《UltraScale+ Integrated Block for PCI Express LogiCORE IP 产品指南》(PG213)

## “GT Settings” 选项卡

“GT 设置 (GT Settings)” 选项卡如下图所示。

图 20: “GT Settings” 选项卡



如需获取这些选项的描述，请参阅下列相应产品指南中的第 4 章节“设计流程步骤”：

- 《7 系列 FPGA Integrated Block for PCI Express LogiCORE IP 产品指南》(PG054)
- 《UltraScale 器件 Gen3 Integrated Block for PCI Express LogiCORE IP 产品指南》(PG156)
- 《UltraScale+ Integrated Block for PCI Express LogiCORE IP 产品指南》(PG213)

## 输出生成

欲知详情，请参阅《Vivado Design Suite 用户指南：采用 IP 进行设计》(UG896)。

## 约束子系统

本节包含与 Vivado® Design Suite 中的子系统约束有关的信息。

### 所需约束

DMA/Bridge Subsystem for PCI Express® 需满足时序约束及其它物理实现约束的规格，方可满足指定的 PCI Express 性能要求。这些约束在赛灵思设计约束 (XDC) 文件中提供。生成的 XDC 中的管脚分配和层级名称对应于所提供的设计示例。



**重要提示！** 如果不使用设计示例顶层文件，则请将参考时钟的 IBUFDS\_GTE3（对应 UltraScale+ IBUFDS\_GTE4）实例、sys\_rst 的 IBUF 实例以及与此些实例关联的位置和时序约束复制到您的本地设计顶层。

为了达成一致的实现结果，通过赛灵思工具运行设计时，必须使用包含这些未经修改的原始约束的 XDC。如需获取有关 XDC 或特定约束的定义及其使用方式的更多详细信息，请参阅《Vivado Design Suite 用户指南：使用约束》(UG903)。

随集成块解决方案提供的约束已通过硬件测试，可提供一致结果。约束可修改，但前提是充分了解每个约束的影响。此外，如果设计背离所提供的约束，则对此类设计不予支持。

### 器件、封装和速度等级选择

XDC 的器件选择部分可将有关设计的目标器件、封装和速度等级的信息告知实现工具。



**重要提示！** 由于第 2 代 (Gen2) 和第 3 代 (Gen3) Integrated Block for PCIe 核都是为特定器件封装组合专门设计的，因此这部分不应修改。

器件选择部分始终包含器件选择行，但也包含特定于器件或封装的选项。以下显示了器件选择行示例：

```
CONFIG PART = XCKU040-ffva1156-3-e-es1
```

### 时钟频率、时钟管理和时钟布局

如需了解有关时钟要求的详细信息，请参阅下列相应的产品指南：

- 《7 系列 FPGA Integrated Block for PCI Express LogiCORE IP 产品指南》(PG054)
- 《Virtex-7 FPGA Integrated Block for PCI Express LogiCORE IP 产品指南》(PG023)
- 《UltraScale 器件 Gen3 Integrated Block for PCI Express LogiCORE IP 产品指南》(PG156)
- 《UltraScale+ Integrated Block for PCI Express LogiCORE IP 产品指南》(PG213)

### bank 分配

本节不适用于此 IP 子系统。

### 收发器布局

本节不适用于此 IP 子系统。

### I/O 标准与布局

本节不适用于此 IP 子系统。

## 调整集成块核的位置

默认情况下，IP 核级约束可将块 RAM、收发器和 PCIe 块锁定到建议的位置。要调整这些块的位置，必须在 XDC 约束文件中覆盖这些块的约束。为此，请执行以下操作：

1. 从核级 XDC 约束文件复制需要覆盖的块的约束。
2. 将这些约束置于用户 XDC 约束文件中。
3. 将约束更新到新位置。

用户 XDC 约束通常限定为设计顶层；因此，请确保这些约束所引用的单元在复制粘贴后仍有效。通常，您需要以完整层级名称来更新模块路径。

**注释：**如果某些位置需要进行交换位置（即，新位置当前被另一个模块占据），有 2 种方式来执行此操作。

- 如有临时位置可用，请首先将第 1 个模块移至新的临时位置。然后，将第 2 个模块移至原先被第 1 个模块占据的位置。下一步，将第 1 个模块移至第 2 个模块的位置。这些步骤可在 XDC 约束文件中完成。
- 如果没有其它位置可用作为临时位置，请在 Tcl 命令窗口中对第 1 个模块使用 `reset_property` 命令，然后将第 2 个模块移至此位置。`reset_property` 命令无法在 XDC 约束文件中执行，必须从 Tcl 命令文件调用，或者直接输入 Tcl 控制台 (Tcl Console)。

## 仿真

本节包含有关在 Vivado® Design Suite 中进行 IP 仿真的信息。

如需获取有关 Vivado® 仿真组件的全面信息以及有关如何使用受支持的第三方工具的信息，请参阅《Vivado Design Suite 用户指南：逻辑仿真》(UG900)。

### 基本仿真

用户可生成对应 AXI-MM 和 AXI-ST 的仿真模型选项并将其用于仿真。这些模型均为非常基本的仿真模型选项，您可在其基础上开发复杂设计。

#### AXI-MM 模式

AXI4 Memory Mapped (AXI-MM) 模式的设计示例在用户侧具有 4 KB 块 RAM，因此可将数据写入此块 RAM，也可将数据从块 RAM 读取到主机。首先启动 H2C 传输，然后 DMA 从主机存储器读取数据，并将其写入块 RAM。随后，启动 C2H 传输，DMA 从块 RAM 读取数据并将其写入主机存储器。原始数据将用于与 C2H 写入数据进行比对。

H2C 和 C2H 各设有 1 个描述符，总计传输大小为 64 个字节。

#### AXI-ST 模式

AXI4-Stream (AXI-ST) 模式的设计示例采用环回设计。在用户侧，H2C 端口环回至 C2H 端口。首先，启动 C2H 传输，C2H DMA 引擎等待用户侧的数据。随后，启动 H2C 传输，DMA 引擎从主机存储器读取数据，并将其写入用户侧。由于这是环回过程，来自 H2C 的设计数据将被转发至 C2H，最终到达主机目标地址。

H2C 和 C2H 各设有 1 个描述符，总计传输大小为 64 个字节。

在 Vivado® Design Suite 仿真中不使用中断。改为轮询描述符完成计数以判定传输完成。

#### 描述符旁路

描述符旁路模式的仿真模型仅可用于通道 0。此设计可扩展以支持其它通道。

### PIPE 模式仿真

DMA/Bridge Subsystem for PCI Express® 支持 PIPE 模式仿真，在此模式下，核的 PIPE 接口连接到链路合作伙伴的 PIPE 接口。此模式可提升仿真速度。

在“端点 (Endpoint)”模式和“根端口 (Root Port)”模式下，均可使用“Customize IP”对话框的“Basic”页面上的“Enable PIPE Simulation”选项在当前 Vivado® Design Suite 解决方案设计示例中启用 PIPE 模式仿真。在核边界处生成的外部 PIPE 接口信号可用于访问外部器件。启用该功能还可提供必要的挂钩，以便使用第三方 PCI Express VIP/BFM 代替随设计示例提供的 Root Port 模型。另请参阅《使用集成端点 PCI Express 块采用 Gen2 x8 和 Gen3 x8 配置进行 PIPE 模式仿真应用指南》(XAPP1184)。

下表描述了核顶层可用的 PIPE 总线信号及其在 EP 核 (pcie\_top) PIPE 信号内的对应映射。



**重要提示！**在仿真目录内提供了 xil\_sig2pipe.v 文件，此文件用于替代 phy\_sig\_gen.v。BFM/VIP 应与 board.v 中的 xil\_sig2pipe 实例相连。

选择 VHDL 作为目标语言时，该核不支持 PIPE 模式仿真。

表 131：常用输入/输出命令和端点 PIPE 信号映射

输入命令	端点 PIPE 信号映射	输出命令	端点 PIPE 信号映射
common_commands_in[25:0]	不使用	common_commands_out[0]	pipe_clk <sup>1</sup>
		common_commands_out[2:1]	pipe_tx_rate_gt <sup>2</sup>
		common_commands_out[3]	pipe_tx_rcvr_det_gt
		common_commands_out[6:4]	pipe_tx_margin_gt
		common_commands_out[7]	pipe_tx_swing_gt
		common_commands_out[8]	pipe_tx_reset_gt
		common_commands_out[9]	pipe_tx_deemph_gt
		common_commands_out[16:10]	不使用 <sup>3</sup>

注释：

1. pipe\_clk 是基于核配置的输出时钟。对于 Gen1 速率，pipe\_clk 为 125 MHz。对于 Gen2 和 Gen3，pipe\_clk 为 250 MHz。
2. pipe\_tx\_rate\_gt 表示流水线速率（2'b00-Gen1、2'b01-Gen2 和 2'b10-Gen3）。
3. 此端口的功能已被弃用，可将其保留并保持未连接状态。

表 132：输入/输出总线与端点 PIPE 信号映射

输入总线	端点 PIPE 信号映射	输出总线	端点 PIPE 信号映射
pipe_rx_0_sigs[31:0]	pipe_rx0_data_gt	pipe_tx_0_sigs[31:0]	pipe_tx0_data_gt
pipe_rx_0_sigs[33:32]	pipe_rx0_char_is_k_gt	pipe_tx_0_sigs[33:32]	pipe_tx0_char_is_k_gt
pipe_rx_0_sigs[34]	pipe_rx0_elec_idle_gt	pipe_tx_0_sigs[34]	pipe_tx0_elec_idle_gt
pipe_rx_0_sigs[35]	pipe_rx0_data_valid_gt	pipe_tx_0_sigs[35]	pipe_tx0_data_valid_gt
pipe_rx_0_sigs[36]	pipe_rx0_start_block_gt	pipe_tx_0_sigs[36]	pipe_tx0_start_block_gt
pipe_rx_0_sigs[38:37]	pipe_rx0_syncheader_gt	pipe_tx_0_sigs[38:37]	pipe_tx0_syncheader_gt
pipe_rx_0_sigs[83:39]	不使用	pipe_tx_0_sigs[39]	pipe_tx0_polarity_gt
		pipe_tx_0_sigs[41:40]	pipe_tx0_powerdown_gt
		pipe_tx_0_sigs[69:42]	不使用 <sup>1</sup>

注释：

1. 此端口的功能已被弃用，可将其保留并保持未连接状态。

## 定制 PIPE 仿真的参数

对于 PIPE 仿真，某些参数是必需的，并且可能需要手动设置。在设计示例中提供了这些必需的参数。从 Vivado IP 目录生成设计实例时，所有必需参数均已设置完成，无需额外操作。但是，定制设计需要您将以下参数添加到自己的设计测试激励文件中。

```
defparam board.AXI_PCIE_EP.xdma_0_i.inst.pcie4_ip_i.inst.PL_SIM_FAST_LINK_TRAINING=2'h3;
localparam EXT_PIPE_SIM = "TRUE";
defparam board.AXI_PCIE_EP.xdma_0_i.inst.pcie4_ip_i.inst.EXT_PIPE_SIM = EXT_PIPE_SIM;
defparam board.RP.pcie_4_0_rport.pcie_4_0_int_inst.EXT_PIPE_SIM = "TRUE";
defparam board.RP.EXT_PIPE_SIM = "TRUE";
```

---

## 综合与实现

如需了解有关综合和实现方面的详情，请参阅《Vivado Design Suite 用户指南：采用 IP 进行设计》([UG896](#))。

# 设计示例

本章包含有关 Vivado<sup>®</sup> Design Suite 中提供的设计示例的信息。

---

## 可用设计示例

设计示例如下所述：

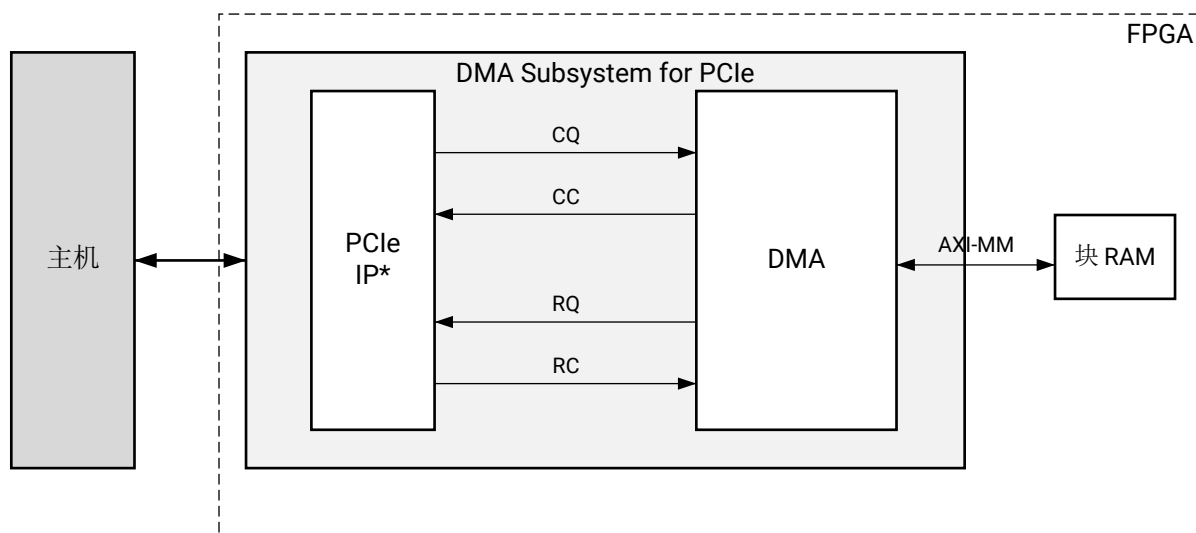
- [AXI4 Memory Mapped 默认设计示例](#)
- [含 PCIe 到 AXI4-Lite 主接口和 PCIe 到 DMA 旁路接口的 AXI4 Memory Mapped 接口设计示例](#)
- [含 AXI4-Lite 从接口的 AXI4 Memory Mapped 设计示例](#)
- [AXI4-Stream 设计示例](#)
- [含描述符旁路的 AXI4 Memory Mapped 示例](#)
- [基于 Vivado IP integrator 的设计示例](#)
- [用户 IRQ 设计示例](#)



## AXI4 Memory Mapped 默认设计示例

下图显示了作为默认设计的 AXI4 Memory Mapped (AXI-MM) 接口。此设计示例在含 AXI4 MM 接口的用户设计上提供了 4KB 的块 RAM。对于 H2C 传输，DMA/Bridge Subsystem for PCI Express® 会从主机读取数据，并写入用户侧的块 RAM。对于 C2H 传输，DMA/Bridge Subsystem for PCI Express® 会从块 RAM 读取数据，并写入主机存储器。来自 IP 目录的设计示例仅含 4 KB 块 RAM；如果需要，您可重新生成子系统以获取更大的块 RAM。

图 21：AXI-MM 默认设计示例



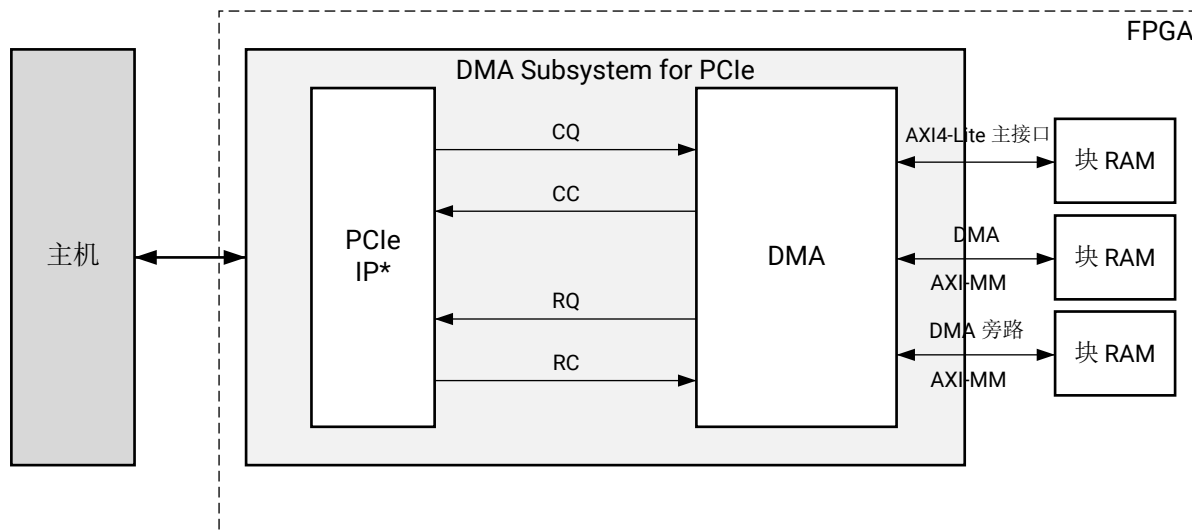
\* 可按需包含封装器

X15052-051621

## 含 PCIe 到 AXI4-Lite 主接口和 PCIe 到 DMA 旁路接口的 AXI4 Memory Mapped 接口设计示例

下图所示系统中，已选中 PCIe 到 AXI4-Lite 主接口 (BAR0) 和 PCIe 到 DMA 旁路 (BAR2) 接口。4K 块 RAM 连接至 PCIe 到 DMA 旁路接口。主机可使用 DMA 旁路接口来通过 AXI4 MM 接口对用户空间读取/写入数据。此接口会绕过 DMA 引擎。主机还可使用 PCIe 到 AXI4-Lite 主接口 (BAR0 地址空间) 来写入/读取用户逻辑。此设计示例将 4K 块 RAM 连接至 PCIe 到 AXI4-Lite 主接口，以便用户应用可执行读写。

图 22：已启用 PCIe 到 DMA 旁路接口和 PCIe 到 AXI-Lite 主接口的 AXI-MM 接口示例



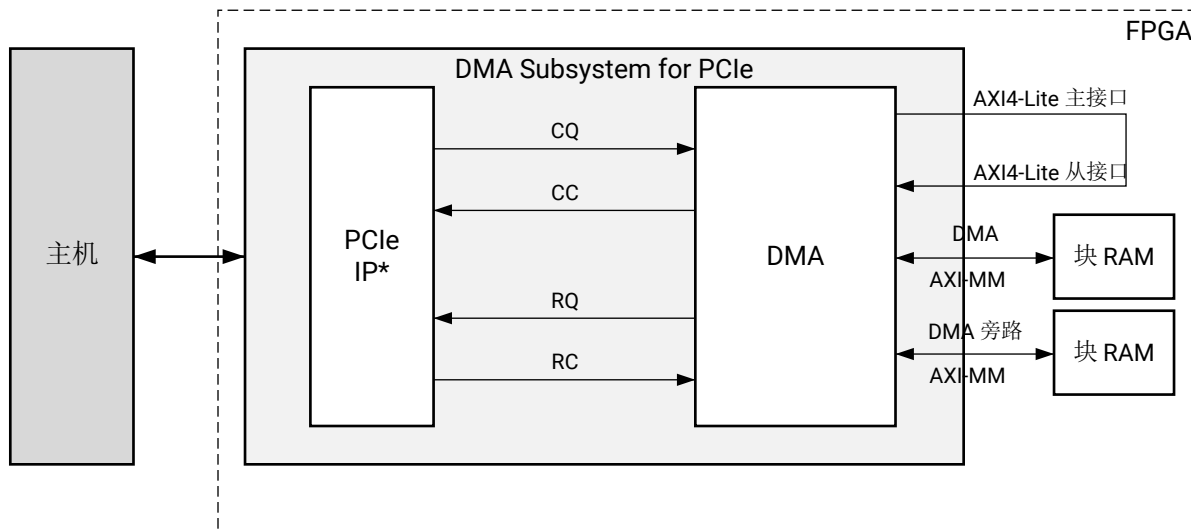
\* 可按需包含封装器

X15047-051621

## 含 AXI4-Lite 从接口的 AXI4 Memory Mapped 设计示例

启用 PCIe® 到 AXI4-Lite 主接口和 AXI4-Lite 从接口时，生成的设计示例（如下图所示）具有从 AXI4-Lite 主接口到 AXI4-Lite 从接口的环回。通常，用户逻辑可以使用 AXI4-Lite 从接口来读写 DMA/Bridge Subsystem for PCI Express® 寄存器。对于此设计示例，主机可以使用 PCIe 到 AXI4-Lite 主接口（BAR0 地址空间）并读写 DMA/Bridge Subsystem for PCI Express® 寄存器，与使用 PCIe 到 DMA（BAR1 地址空间）相同。此设计示例还可显示启用 PCIe 到 DMA 旁路接口（BAR2）情况下的示例。

图 23：启用 AXI-Lite 从接口的 AXI-MM 示例



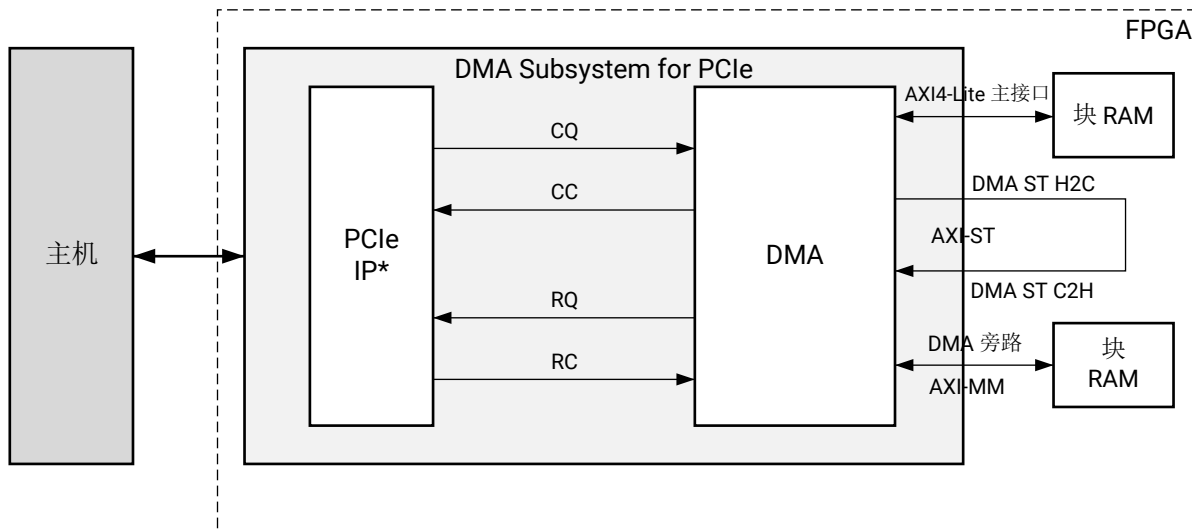
\* 可按需包含封装器

X15045-051621

## AXI4-Stream 设计示例

启用 AXI4-Stream 接口时，每条 H2C 数据流传输通道都环回至 C2H 通道。如下图所示，设计示例提供了适用于 AXI4 Streaming 的环回设计。其局限性在于所选 H2C 通道和 C2H 通道的数量必须相同才能正常工作。此设计示例还演示了选中 PCIe 到 DMA 旁路接口以及 PCIe 到 AXI-Lite 主接口情况下的示例。

图 24：已启用 PCIe 到 DMA 旁路接口和 PCIe 到 AXI-Lite 主接口情况下的 AXI4-Stream 示例



\* 可按需包含封装器

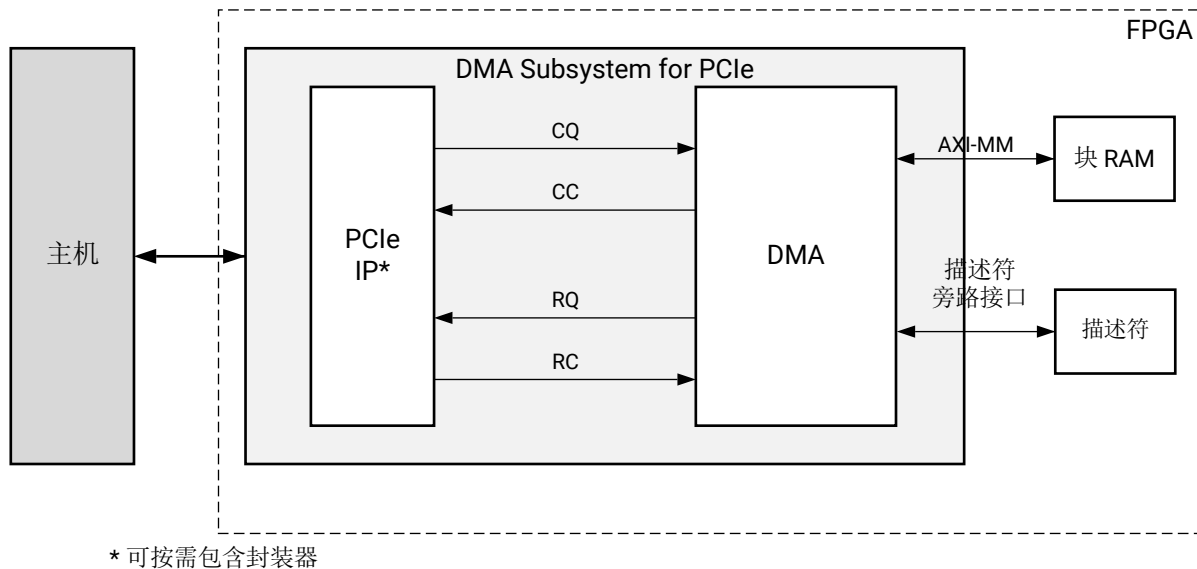
X15046-051621

## 含描述符旁路的 AXI4 Memory Mapped 示例

启用描述符旁路 (Descriptor bypass) 模式后，用户逻辑负责生成描述符并在描述符旁路接口中传输这些描述符。下图显示了启用描述符旁路模式的 AXI4 Memory Mapped 设计。您可选择采用描述符旁路模式的通道。针对描述符旁路模式选中 H2C 的通道 0 和 C2H 的通道 0 时，生成的 Vivado® 设计示例会将其 H2C0 和 C2H0 的描述符旁路端口连接到仅生成单一描述符（64 个字节）的逻辑。用户负责为其它通道开发代码和扩展描述符本身。

下图显示了启用描述符旁路模式的 AXI-MM 示例。

图 25：启用描述符旁路模式的 AXI-MM 示例



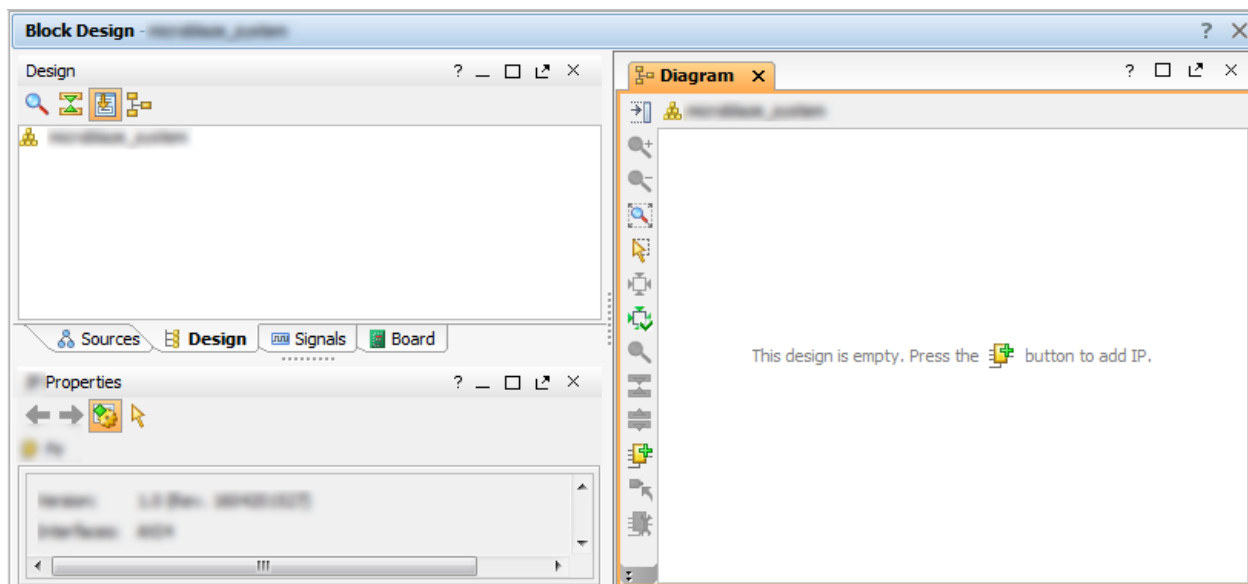
X17931-051621

## 基于 Vivado IP integrator 的设计示例

除了基于 RTL 的设计示例外，IP 还支持基于 Vivado® IP integrator 的设计示例。要使用的设计示例，请执行以下操作：

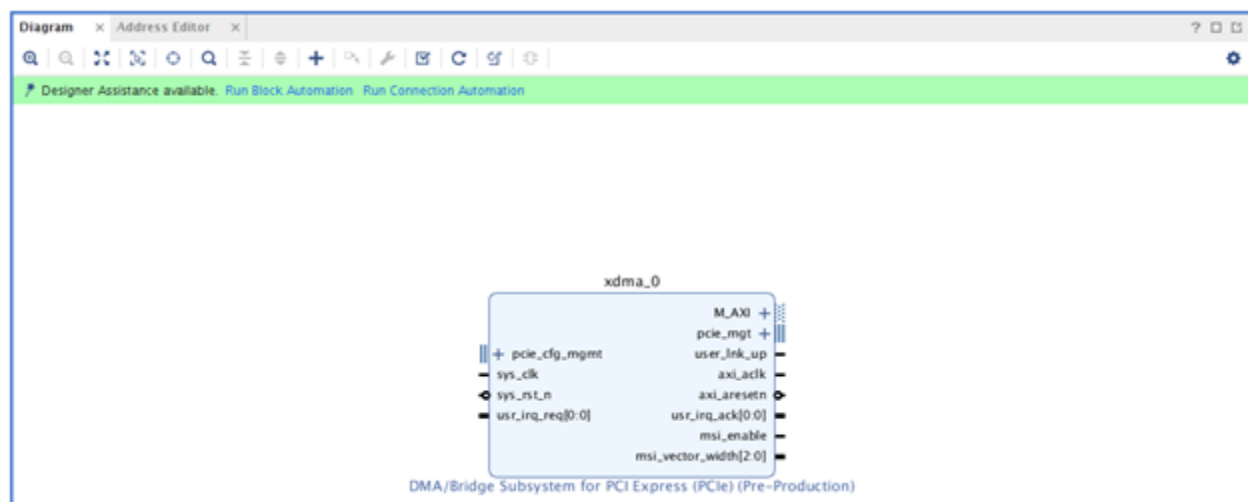
1. 创建 IP integrator 模块框图。
2. 打开 IP integrator 工作空间，如下图所示。

图 26: Vivado IP integrator 初始视图，其中显示了 1 条参考消息



3. 要在画布中添加 DMA/Bridge IP，请在 IP 目录中搜索 DMA/Bridge (xdma) IP。  
在画布中添加此 IP 后，就会在画布顶部显示绿色“设计辅助 (Designer Assistance)”信息栏。

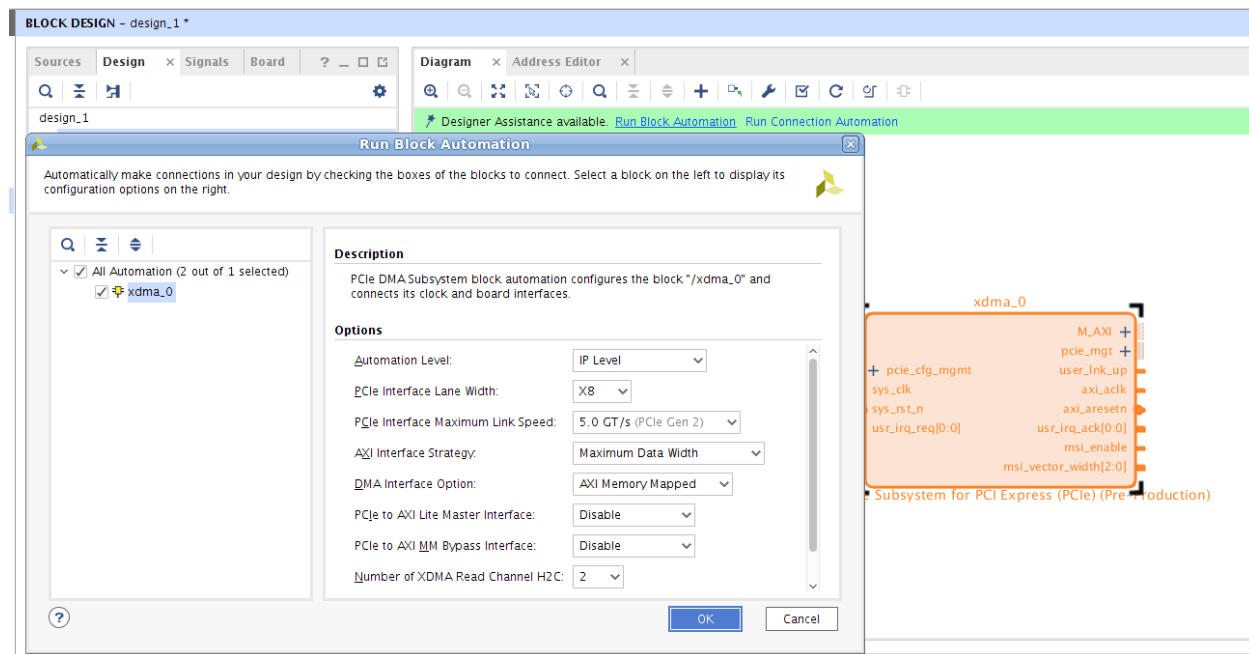
图 27: 可提供块自动化设置的 Designer Assistance



4. 单击 Designer Assistance 信息栏中的“Run Block Automation”。

这样即可打开“运行块自动化设置 (Run Block Automation)”对话框（如下图所示），其中列出了当前设计中可执行块自动化设置的所有 IP（左侧窗格），以及与特定自动化设置相关联的任意选项（右侧窗格）。在此情况下，左侧窗格中的层级中只有 XDMA IP。右侧窗格包含描述以及可用选项。“选项 (Options)”可用于配置 IP 以及判断块自动化设置的自动化级别。

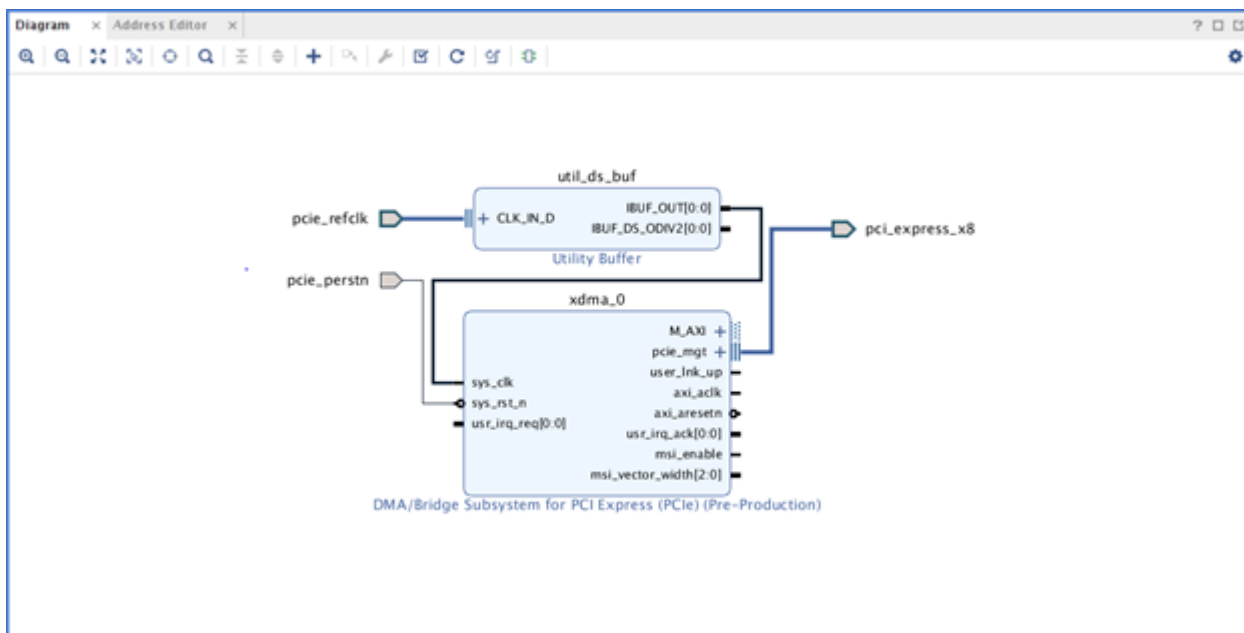
图 28：“Run Block Automation”对话框



“Run Block Automation”对话框包含“自动化级别 (Automation Level)”选项，该选项可设置为“IP 级别 (IP Level)”或“子系统级别 (Subsystem Level)”。

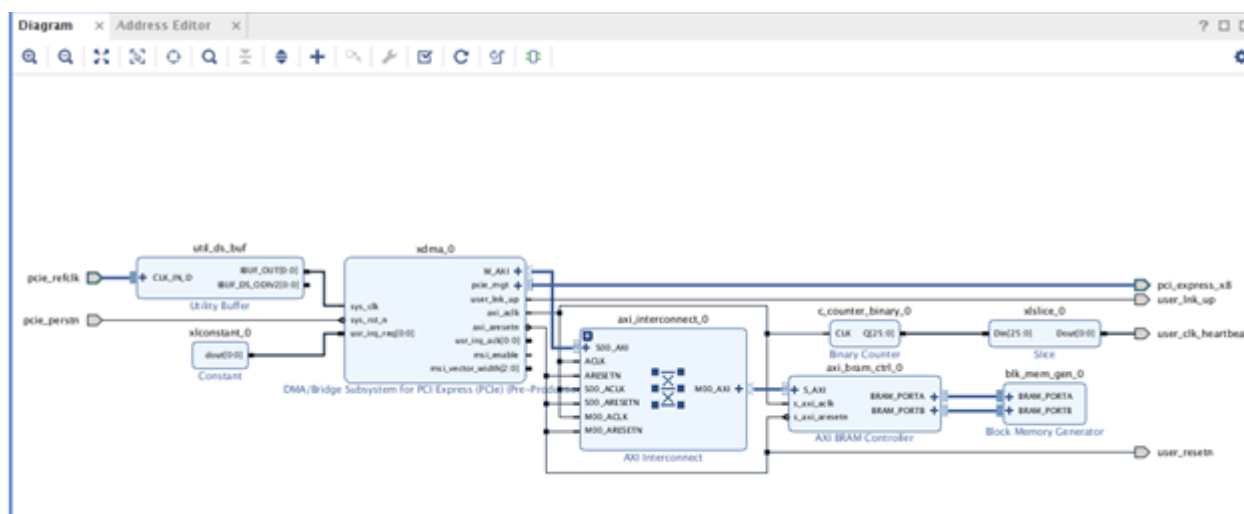
- IP Level: 选择 IP 级别自动化时，“Block Automation”会为 `sys_clk` 输入插入实用工具缓冲器，并为 XDMA IP 连接 `sys_rst_n` 输入和 `pcie_mgt` 输出接口，如下图所示。

图 29：IP 级别块自动化设置



- Subsystem Level: 选择子系统级别自动化时，“Block Automation”会在画布上插入必要的子 IP 并建立必要的连接。除了连接 `sys_clk` 和 `sys_rst_n` 输入外，它还会连接 `pcie_mgt` 输出接口，并连接 `user_lnk_up`、`user_clk_heartbeat` 和 `user_resetn` 输出。它会插入 AXI Interconnect 以通过 AXI BRAM 控制器来将块存储器 (Block Memory) 与 XDMA IP 相连。在“Run Block Automation”对话框中启用 AXI4-Lite 主接口和 AXI-MM 旁路接口时，AXI Interconnect 具有 1 个主接口和多个从接口。启用 AXI4-Lite 主接口和 AXI-MM 旁路接口时，此块自动化还会插入块存储器和 AXI BRAM 控制器。

图 30：子系统级别块自动化设置





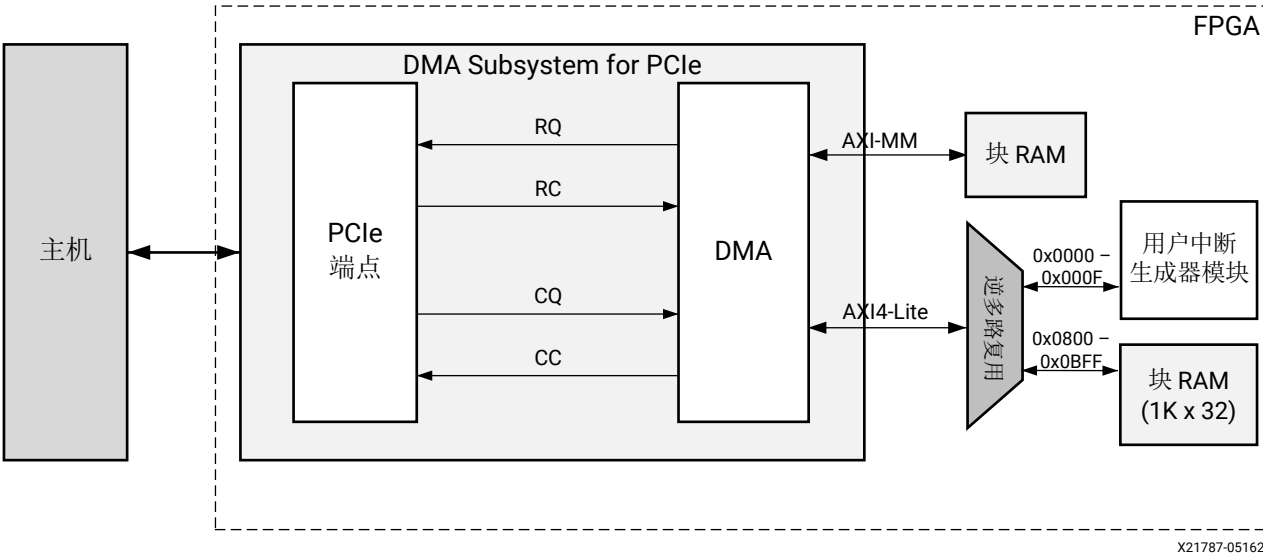
用户 IRQ 设计示例

用户 IRQ 设计示例支持采用默认 DMA/Bridge Subsystem for PCI Express® 设计示例将主机连接到 AXI4-Lite 主接口。在设计示例中，AXI4-Lite 接口上集成了用户中断生成器模块和外部块 RAM。主机可使用此接口通过写入用户中断生成器模块的寄存器空间来生成用户 IRQ，还可对外部 1K 块 RAM 执行读取/写入。下图显示了设计示例。

设计示例可使用以下 Tcl 命令来生成。

```
set_property -dict [list CONFIG.usr_irq_exdes {true}] [get_ips <ip_name>]
```

图 31：用户 IRQ 设计示例



下表中提供了寄存器描述。

表 133：设计示例寄存器

寄存器偏移	寄存器名称	访问类型	描述
0x00	Scratch Pad	RW	暂存寄存器
0x04	DMA BRAM Size	RO	连接到 XDMA 的用户存储器大小。 存储器大小 = (2[7:4]) ([3:0]Byte) [7:4] - 以 2 次幂来表示大小。 0 - 1 1 - 2 2 - 4 ... 8 - 256 9 - 512 [3:0] - 表示单位。 0 - 字节 1 - KB 2 - MB 3 - GB 例如，如果寄存器值为 21，大小为 4 KB。如果寄存器值为 91，大小为 512 KB。

表 133：设计示例寄存器 (续)

寄存器偏移	寄存器名称	访问类型	描述
0x08	Interrupt Control Register	RW	中断控制寄存器（写入 1 以生成中断）。 中断状态寄存器对应位必须为 1（就绪）才能生成中断。并且，提供 ISR 后，将对应的位复位。
0x0C	Interrupt Status Register	RO	中断状态。 1：就绪 0：正在进行中断生成

**注释：**对于传统中断，只有在提供 ISR 之后才应清除对应中断位的中断控制寄存器 (0x08) 值，因为这可供主机用于判定中断源。

## 自定义和生成设计示例

在“自定义 IP (Customize IP)”对话框中，使用 IP 设计示例的默认参数值。

复查 IP 参数后，请执行以下操作：

1. 右键单击组件名称。
2. 选择“Open IP Example Design”。

这样即可打开独立的设计示例。

# 测试激励文件

本章包含有关 Vivado® Design Suite 中提供的测试激励文件的信息。

## 对应端点的根端口模型测试激励文件

PCI Express® 根端口模型是基本的测试激励文件环境，可提供测试程序接口，此接口可配合提供的 PIO 设计或您的设计一起使用。根端口模型的用途是提供源机制用于生成下游 PCI Express TLP 流量以对客户设计进行仿真，并提供目标机制用于接收来自仿真环境内的客户设计的上游 PCI Express TLP 流量。其中包含的根端口模型的源代码可提供模型，作为您的测试激励文件的起点。包括初始化核配置空间、创建 TLP 传输事务、生成 TLP 日志以及提供接口用于创建和验证测试在内的所有重要工作都在其中完成，使您能够集中精力验证设计功能是否正确，而无需将时间用于开发端点核的测试激励文件基础架构。

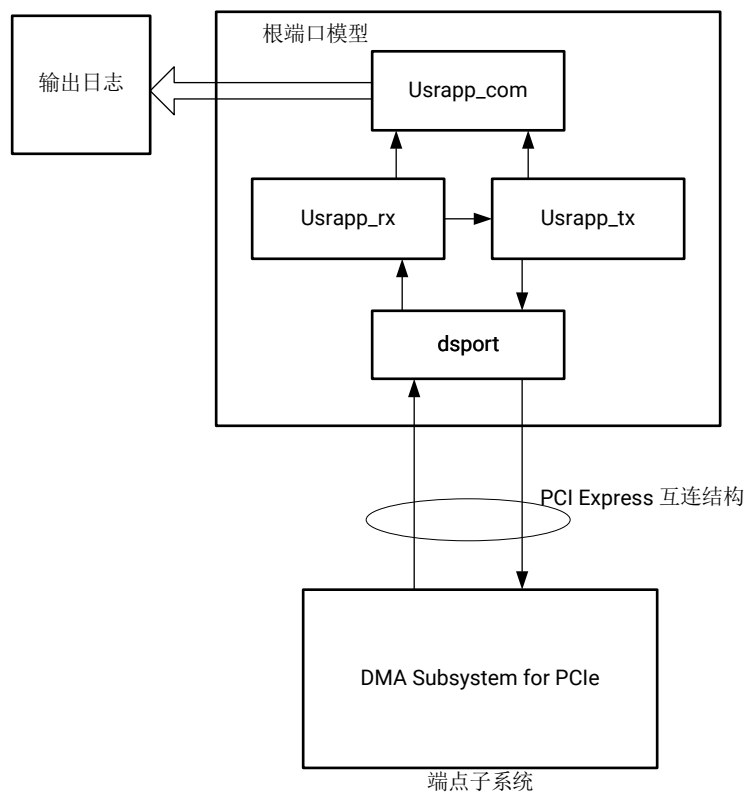
其中包含的根端口模型的源代码可提供模型，作为您的测试激励文件的起点。包括初始化核配置空间、创建 TLP 传输事务、生成 TLP 日志以及提供接口用于创建和验证测试在内的所有重要工作都在其中完成，使您能够集中精力验证设计功能是否正确，而无需将时间用于开发端点核的测试激励文件基础架构。

根端口模型包括：

- 测试编程接口 (TPI)，支持您对 PCI Express 的端点器件进行仿真。
- 测试示例，用于演示如何使用测试程序的 TPI。
- Verilog 源代码，用于所有根端口模型组件，以支持您对测试激励文件进行自定义。

下图显示了 DMA Subsystem for PCIe 的根端口模块。

图 32：DMA Subsystem for PCIe 的根端口模块



X15051-051621

## 架构

上图所示的“根端口 (Root Port)”模型包含下列块：

- dsport（根端口）
- usrcpp\_tx
- usrcpp\_rx
- usrcpp\_com（仅限 Verilog）

usrcpp\_tx 和 usrcpp\_rx 块与 dsport 块相连，以通过端点 DUT 发射和接收 TLP。端点 DUT 包含 DMA Subsystem for PCIe。

usrcpp\_tx 块将 TLP 发送至 dsport 块，以便跨 PCI Express 链路发送至端点 DUT。而端点 DUT 器件则通过 PCI Express 链路将 TLP 发射至 dsport 块，随后此块将被传递到 usrcpp\_rx 块。通过 PCI Express 逻辑进行通信时，dsport 与核共同负责数据链路层和物理链路层处理。usrcpp\_tx 和 usrcpp\_rx 均使用 usrcpp\_com 来执行共享功能，例如，TLP 处理和日志文件输出。

PIO 写入和读取由 usrcpp\_tx 进行初始化。

DMA Subsystem for PCIe 使用 7 系列 Gen2 Integrated Block for PCIe、7 系列 Gen3 Integrated Block for PCIe、UltraScale™ 器件 Gen3 Integrate Block for PCIe 和 UltraScale+™ 器件 Integrate Block for PCIe。请参阅相应指南中的“测试激励文件”章节：

- 《7 系列 FPGA Integrated Block for PCI Express LogiCORE IP 产品指南》(PG054)

- 《Virtex-7 FPGA Integrated Block for PCI Express LogiCORE IP 产品指南》(PG023)
- 《UltraScale 器件 Gen3 Integrated Block for PCI Express LogiCORE IP 产品指南》(PG156)
- 《UltraScale+ Integrated Block for PCI Express LogiCORE IP 产品指南》(PG213)

## 测试案例

DMA Subsystem for PCIe 可配置为 AXI4 Memory Mapped (AXI-MM) 或 AXI4-Stream (AXI-ST) 接口。仿真测试案例会读取配置寄存器以判定采用的是 AXI4 Memory Mapped 还是 AXI4-Stream 配置。测试案例会根据 AXI 设置对任一配置执行仿真。

表 134：测试案例描述

测试案例名称	描述
Dma_test0	AXI4 Memory Mapped 接口仿真。从主机存储器读取数据，并将其写入块 RAM (H2C)。然后，从块 RAM 读取数据，并将其写入主机存储器 (C2H)。最后，测试案例会比较数据，确认是否正确。
Dma_stream0	AXI4-Stream 接口仿真。从主机存储器读取数据并发送至 AXI4-Stream 用户接口 (H2C)，数据会环回至主机存储器 (C2H)。

## 仿真

仿真设置为在 H2C 中传输 1 个描述符，在 C2H 方向也传输 1 个描述符。在每个描述符中，传输大小设置为 128 个字节。对于 AXI-MM 和 AXI-Stream，则从主机读取数据，并将其发送到卡 (H2C)。随后，从卡读取数据并发送至主机 (C2H)。从卡读取的数据将与原始数据比对以验证数据有效性。

限制：

- 仿真不支持中断。测试案例仅读取状态和完成描述符计数寄存器以判定传输是否完成。
- 仅对通道 0 执行仿真。在未来版本中，将启用多通道仿真。
- 传输大小限制为 128 个字节，且仅限 1 个描述符。
- 根端口仿真模型并非完整的 BFM。仿真支持单描述符传输，该传输显示基本 DMA 流程。
- 默认情况下，设计示例不支持综合后仿真。要启用综合后仿真，请使用以下 Tcl 命令生成 IP：

```
set_property -dict [list CONFIG.post_synth_sim_en {true}] [get_ips <ip_name>]
```

**注释：**使用此功能时，功能仿真时间可能增加至约 2.5 ms。

## AXI4 Memory Mapped 接口

首先，测试案例启动 H2C 引擎。H2C 引擎会从主机存储器读取数据，并将其写入用户侧的块 RAM。随后，测试案例会启动 C2H 引擎。C2H 引擎会从块 RAM 读取数据，并将其写入主机存储器。仿真步骤如下所示：

1. 测试案例为 H2C 引擎设置 1 个描述符。
2. 在主机存储器中创建 H2C 描述符。H2C 描述符提供数据长度（128 个字节）、源地址（主机）和目标地址（卡）。
3. 测试案例在源地址空间内写入数据（128 个字节的增量数据）。
4. 测试案例还会为 C2H 引擎设置 1 个描述符。

5. C2H 描述符提供数据长度（128 个字节）、源地址（卡）和目标地址（主机）。
6. 将 H2C 描述符起始地址写入寄存器（0x4080 和 0x4084）。
7. 写入 H2C 控制寄存器，这样即可启动 H2C 传输地址（0x0004）。位 0 设为 1 即可启动传输。如需了解有关控制寄存器的详细信息，请参阅 H2C Channel Control (0x04)。
8. DMA 传输会从主机源地址提取数据，并将其发送至块 RAM 目标地址。
9. 随后，测试案例会启动 C2H 传输。
10. 将 C2H 描述符起始地址写入寄存器（0x5080 和 0x5084）。
11. 写入 C2H 控制寄存器，这样即可启动 C2H 传输（0x1004）。位 0 设为 1 即可启动传输。如需了解有关控制寄存器的详细信息，请参阅 C2H Channel Control (0x04)。
12. DMA 传输会从块 RAM 源地址提取数据，并将此数据发送至主机目标地址。
13. 随后，测试案例会比较数据，确认是否正确。
14. 测试案例会检查 H2C 和 C2H 描述符的完成计数（值为 1）。
15. 随后，测试案例会停用 H2C 和 C2H 引擎的控制寄存器 0x0004 和 0x1004 中的 Run 位（位 0），以禁用传输。

## AXI4-Stream 接口

对于 AXI4-Stream，设计示例为环回设计。对于所有其它通道，通道 H2C\_0 环回至 C2H\_0（以此类推）。首先，测试案例启动 C2H 引擎。C2H 引擎会等待由 H2C 引擎发射的数据。随后，测试案例会启动 H2C 引擎。H2C 引擎会从主机读取数据并将其发送至卡，随后此数据会环回至 C2H 引擎。然后，C2H 引擎会将此数据写回主机存储器。仿真步骤如下所示：

1. 测试案例为 H2C 引擎设置 1 个描述符。
2. 在主机存储器中创建 H2C 描述符。H2C 描述符提供数据长度（128 个字节）、源地址（主机）和目标地址（卡）。
3. 测试案例在主机源地址空间内写入数据（128 个字节的增量数据）。
4. 测试案例还会在主机存储器中为 C2H 引擎设置 1 个描述符。
5. C2H 描述符提供数据长度（128 个字节）、源地址（卡）和目标地址（主机）。
6. 将 C2H 描述符起始地址写入寄存器（0x5080 和 0x5084）。
7. 写入 C2H 控制寄存器，这样即可首先启动 C2H 传输。
8. C2H 引擎将启动并等待来自 H2C 端口的数据。
9. 将 H2C 描述符起始地址写入寄存器（0x4080 和 0x4084）。
10. 写入 H2C 控制寄存器，这样即可启动 H2C 传输。
11. H2C 引擎会将来自主机源地址的数据发送至卡目标地址。
12. 此数据将环回至 C2H 引擎。
13. C2H 引擎会从卡读取数据，并将其写回主机存储器目标地址。
14. 测试案例会检查 H2C 和 C2H 描述符的完成计数（值为 1）。
15. 随后，测试案例会比较数据，确认是否正确。
16. 随后，测试案例会停用 H2C 和 C2H 引擎的控制寄存器 0x0004 和 0x1004 中的 Run 位（位 0），以禁用传输。

## 描述符旁路模式

当针对描述符旁路选项同时选中 H2C 和 C2H 的通道 0 时，即可对描述符旁路模式执行仿真。生成的设计示例有 1 个处于就绪状态的描述符用于接入描述符旁路模式接口。

### AXI-MM 描述符旁路模式仿真

1. 设计示例具有对应 H2C 和 C2H 引擎的预定义描述符。
2. H2C 描述符包含 128 个字节的数据、源地址（主机）和目标地址（卡）。
3. C2H 描述符包含 128 个字节的数据、源地址（卡）和目标地址（主机）。
4. 测试案例会向主机存储器源地址写入 128 个字节的增量数据。
5. PIO 会写入 H2C 引擎控制寄存器以启动传输 (0x0004)。
6. DMA 会从主机地址读取数据，并将其发送到卡的块 RAM 目标地址。
7. PIO 会写入 C2H 引擎控制寄存器以启动传输 (0x1004)。
8. DMA 会从卡的块 RAM 源地址读取数据，并将其发送到主机目标地址。
9. 测试案例会比较数据，确认是否正确。
10. 测试案例会检查 H2C 和 C2H 描述符的完成计数（值为 1）。
11. 随后，测试案例会通过将 H2C 和 C2H 引擎的控制寄存器 0x0004 和 0x1004 中的 Run 位（位 0）断言无效，以禁用传输。

### 含环回设计的 AXI-Stream 描述符旁路模式仿真

1. 设计示例具有对应 H2C 和 C2H 引擎的预定义描述符。
2. H2C 描述符包含 128 个字节的数据、源地址（主机）和目标地址（卡）。
3. C2H 描述符包含 128 个字节的数据、源地址（卡）和目标地址（主机）。
4. 测试案例会向主机存储器源地址写入 128 个字节的增量数据。
5. PIO 会写入 C2H 引擎控制寄存器以启动传输 (0x1004)。
6. C2H 引擎会启动 DMA 传输，但等待数据（环回设计）。
7. PIO 会写入 H2C 引擎控制寄存器以启动传输 (0x0004)。
8. H2C 引擎会从主机地址读取数据，并将其发送到卡。
9. 此数据将环回至 C2H 引擎。
10. C2H 引擎会从卡读取数据，并将其发送到主机目标地址。
11. 测试案例会比较数据，确认是否正确。
12. 测试案例会检查 H2C 和 C2H 描述符的完成计数（值为 1）。
13. 随后，测试案例会通过将 H2C 和 C2H 引擎的控制寄存器 0x0004 和 0x1004 中的 Run 位（位 0）断言无效，以禁用传输。

当传输开始后，通过描述符旁路接口传输 1 个 H2C 描述符和 1 个 C2H 描述符，然后按上文所述方式执行 DMA 传输。仅针对 64 个字节的传输设置描述符。

## 仿真更新

下文概述了如何修改现有根端口任务来实践多通道案例和多描述符案例。

### 多通道仿真，通道 1 H2C 和 C2H 示例

1. 在主机存储器地址内创建 H2C 通道 1 描述符，该描述符不同于 H2C 和 C2H 通道 0 描述符。
2. 在主机存储器地址内创建 C2H 通道 1 描述符，该描述符不同于 H2C 通道 0、C2H 通道 0 以及 H2C 通道 1 描述符。
3. 在主机存储器内为 H2C 通道 1 传输创建传输数据（128 个字节）不会覆盖主机存储器中的 4 个描述符（H2C 和 C2H 的通道 0 及通道 1 的描述符）中的任一描述符，也不会覆盖 H2C 通道 0 数据。
4. 还需确保主机存储器中的 H2C 数据不会与 C2H 通道 0 和 1 的 C2H 数据传输空间重叠。
5. 将描述符起始地址写入 H2C 通道 0 和 1。
6. 写入 H2C 通道 0 和 1 的控制寄存器（位 0）以启用多通道传输。
7. 写入 C2H 通道 0 和 1 的控制寄存器（位 0）以启用多通道传输。
8. 比较数据，确认是否正确。

此过程同样适用于 AXI-Stream 配置。请参阅上一章节内容以获取 AXI-Stream 传输的详细解释。

### 多描述符仿真

1. 创建含 256 个字节数据（增量数据或任意数据）的传输。将此数据拆分为 2 个数据段（各 128 个字节）。首先，此数据从地址 S1 开始，其次，128 个字节从地址 S2 开始。
2. 在位于 DSC1 处的主机存储器地址中创建新的描述符（名为 DSC\_H2C\_1）。
3. DSC\_H2C\_1 描述符包含对应 DMA 传输、主机地址 S1（源地址）和目标地址 D1（卡）的 128 个字节。
4. 在位于地址 DSC2（不同于 DSC\_H2C\_1 描述符）处的主机存储器中创建新的描述符（名为 DSC\_H2C\_2）。
5. DSC\_H2C\_2 描述符包含对应 DMA 传输、主机地址 S2（源地址）和目标地址 D2（卡）的 128 个字节。
6. 在 DSC\_H2C\_1 中添加下一个描述符地址，以链接这 2 个描述符。在下一个描述符字段中写入 DSC2。
7. 将描述符起始地址连线到 H2C 通道 0。
8. 在控制寄存器 0x0004 中写入运行位，为 H2C 通道 0 启用 DMA 传输。

## 测试任务

表 135：测试任务

名称	描述
TSK_INIT_DATA_H2C	此任务用于为 H2C 引擎生成 1 个描述符，并在主机存储器中初始化源数据。
TSK_INIT_DATA_C2H	此任务用于为 C2H 引擎生成 1 个描述符。
TSK_XDMA_REG_READ	此任务用于读取 DMA Subsystem for PCIe 寄存器。
TSK_XDMA_REG_WRITE	此任务用于写入 DMA Subsystem for PCIe 寄存器。
COMPARE_DATA_H2C	此任务用于将主机存储器中的源数据与写入块 RAM 的目标数据进行比对。此任务在 AXI4 Memory Mapped 仿真中使用。
COMPARE_DATA_C2H	此任务用于将主机存储器中的原始数据与 C2H 引擎写入主机的数据进行比对。此任务在 AXI4 Memory Mapped 仿真中使用。



表 135：测试任务 (续)

名称	描述
TSK_XDMA_FIND_BAR	此任务用于在已启用的不同（BAR0 到 BAR6）之间查找 XDMA 配置空间。

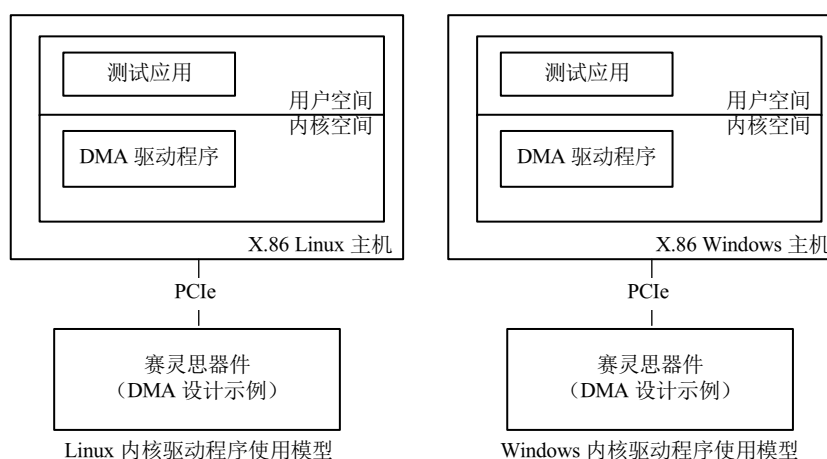
如需了解其它 PCIe 相关任务，请参阅《7 系列 FPGA Integrated Block for PCI Express LogiCORE IP 产品指南》(PG054)、《Virtex-7 FPGA Integrated Block for PCI Express LogiCORE IP 产品指南》(PG023)、《UltraScale 器件 Gen3 Integrated Block for PCI Express LogiCORE IP 产品指南》(PG156) 或《UltraScale+ Integrated Block for PCI Express LogiCORE IP 产品指南》(PG213) 中的“测试激励文件”章节。

# 应用软件开发

本节提供了有关随核一并提供的 Linux 器件驱动程序和 Windows 驱动程序专区的详细信息。

## 器件驱动程序

图 33: 器件驱动程序



X24822-051621

上图显示了 Linux 和 Windows XDMA 软件驱动程序的使用模型。DMA/Bridge Subsystem for PCIe 设计示例是在通过 PCI Express 连接到 X86 主机的赛灵思 FPGA 上实现的。

- 在第一种使用模式中，内核空间中的 XDMA 驱动程序在 Linux 上运行，而测试应用则在用户空间内运行。
- 在第二种使用模式中，内核空间中的 XDMA 驱动程序在 Windows 上运行，而测试应用则在用户空间内运行。

## Linux 器件驱动程序

Linux 器件驱动程序具有以下字符器件接口：

- 用户字符器件，用于访问用户组件。
- 控制字符器件，用于控制 DMA/Bridge Subsystem for PCI Express® 组件。
- 事件字符器件，用于等待中断事件。

- SGDMA 字符器件，用于执行高性能传输。

用户可访问器件如下：

- XDMA0\_control：用于访问 DMA/Bridge Subsystem for PCI Express® 寄存器。
- XDMA0\_user：用于访问 AXI-Lite 主接口。
- XDMA0\_bypass：用于访问 DMA 旁路接口。
- XDMA0\_events\_\*：用于识别用户中断。

## 使用驱动程序

XDMA 驱动程序可从[赛灵思 DMA IP 驱动程序页面](#)下载。

## 中断处理

### 传统中断

传统中断有 4 种类型：A、B、C 和 D。您可在“PCIe Misc”选项卡中的“Legacy Interrupt”设置下选择任意一种类型的中断。您必须对 IRQ 块通道矢量（请参阅“IRQ Block Channel Vector Number (0xA0)”）和 IRQ 块用户矢量（请参阅“IRQ Block User Vector Number (0x80)”）的对应值进行编号。每个传统中断的值为：A = 0、B = 1、C = 2 和 D = 3。主机仅基于这些值来识别中断。

### MSI 中断

对于 MSI 中断，在“PCIe Misc”选项卡中的“MSI Capabilities”下可选 1 到 32 个矢量，“MSI 功能 (MSI Capabilities)”由最多 16 个可用 DMA 中断矢量和最多 16 个可用用户中断矢量组成。Linux 操作系统 (OS) 支持 1 个矢量。其它操作系统可能支持更多矢量，您可在“IRQ 块通道矢量”（请参阅“IRQ Block Channel Vector Number (0xA0)”）和“IRQ 块用户矢量”（请参阅“IRQ Block User Vector Number (0x80)”）中对不同矢量值进行编程，以表示不同中断源。赛灵思 Linux 驱动程序仅支持 1 个 MSI 矢量。

### MSI-X 中断

对于 MSI-X，DMA 支持最多 32 个不同中断源，MSI-X 由最多 16 个可用 DMA 中断矢量和最多 16 个可用用户中断矢量组成。DMA 具有 32 个 MSI-X 表，每个表对应 1 个源（请参阅“MSI-X 矢量和 PBA (0x00-0xFE0)”）。要执行 MSI-X 通道中断处理，驱动程序应使用对应 H2C 和 C2H 的引擎中断使能掩码（请参阅“H2C Channel Interrupt Enable Mask (0x90)”或“C2H Channel Interrupt Enable Mask (0x90)”表）来禁用和启用中断。

### 用户设置

用户逻辑必须使 `usr_irq_req` 保持处于高电平有效状态（即使接收到 `usr_irq_ack` (acks) 后也是如此），以使中断暂挂寄存器保持断言有效。这样即可支持驱动程序内的中断服务例程 (ISR) 判定中断源。当驱动程序接收用户中断后，驱动程序或软件即可通过断言 `usr_irq_req` 无效来将硬件响应的用户中断源复位。

## H2C 流程示例

在 H2C 流程示例中，`loaddriver.sh` 会为所有可用通道加载器件。`dma_to_device` 用户程序用于将数据从主机传输到卡。

H2C 流程示例顺序如下：

1. 打开 H2C 器件并初始化 DMA。
2. 用户程序会读取数据文件、分配缓冲器指针并将该指针随特定器件 (H2C) 和数据大小一起传递到写入函数。
3. 驱动程序会基于输入数据/大小创建描述符，并以描述符起始地址和任何相邻描述符（如有）来初始化 DMA。
4. 驱动程序写入控制寄存器，以启动 DMA 传输。
5. DMA 从主机读取描述符，并开始处理每个描述符。
6. DMA 从主机提取数据并将数据发送至用户侧。基于设置完成所有数据的传输后，DMA 会向主机生成中断。
7. ISR 驱动程序会处理中断以确定正在发送中断的引擎，并检查状态以查看是否存在任何错误。它还会检查正在处理的描述符数量。
8. 状态正常后，驱动会将传输字节长度返回至用户侧，以便在用户侧执行相同的检查。

## C2H 流程示例

在 C2H 流程示例中，`loaddriver.sh` 会为所有可用通道加载器件。`dma_from_device` 用户程序用于将数据从卡传输到主机。

C2H 流程示例顺序如下：

1. 打开器件 C2H 并初始化 DMA。
2. 用户程序会分配缓冲器指针（基于大小）、将该指针传递到具有特定器件 (C2H) 和数据大小的读取函数。
3. 驱动程序会基于大小创建描述符，并使用描述符起始地址初始化 DMA。并且如果存在相邻描述符，
4. 驱动程序会写入控制寄存器，以启动 DMA 传输。
5. DMA 会从主机读取描述符，并开始处理每个描述符。
6. DMA 会从卡提取数据并将此数据发送到主机。基于设置完成所有数据的传输后，DMA 会向主机生成中断。
7. ISR 驱动程序会处理中断以确定正在发送中断的引擎，并检查状态以查看是否存在任何错误以及已处理的描述符数量。
8. 状态正常后，驱动会将传输字节长度返回至用户侧，以便在用户侧执行相同的检查。

# 升级

本附录包含有关升级至更新版本的 IP 的信息。

## 新参数

在当前版本的 IP 中新增了下列新参数。

表 136: 新参数

名称	显示名称	描述	默认值
shared_logic_gtc_7xG2	Include Shared Logic (Transceiver GT_COMMON) in example design	选中此参数时，在设计示例中包含 GT_COMMON 块。	False
shared_logic_clk_7xG2	Include Shared Logic (Clocking) in example design	选中此参数时，在设计示例中包含时钟块。	False
shared_logic_both_7xG2	Include Shared Logic in core	选中此参数时，在核中包含 GT_COMMON 和时钟块。	False
ecc_en	Enable ECC	启用 ECC。需开启任一奇偶校验选项。	False
aspm_support	ASPM Support optionality	表示启用或禁用 ASPM 支持。	No_ASPM

## 新增端口

在 7 系列 Gen2 器件的“Shared Logic”选项卡下选中“Internal Shared GT\_COMMON and Clocking”选项时，下表中的端口就会出现在边界处。

表 137: 适用于共享逻辑的端口（“Internal Shared GT\_COMMON and Clocking”选项）

名称	方向	宽度
int_dclk_out	输出	1 位
int_oobclk_out	输出	1 位
int_pclk_sel_slave	输入	1 位
int_pclk_out_slave	输出	1 位
int_pipe_rxusrclk_out	输出	1 位
int_qplllock_out	输出	2 位
int_qplloutclk_out	输出	2 位
int_qplloutrefclk_out	输出	2 位

表 137: 适用于共享逻辑的端口（“Internal Shared GT\_COMMON and Clocking”选项）（续）

名称	方向	宽度
int_rxoutclk_out	输出	1 位
int_userclk1_out	输出	1 位
int_userclk2_out	输出	1 位

在 7 系列 Gen2 器件的“Shared Logic”选项卡下选中“Shared GT\_COMMON”选项时，下表中的端口就会出现在边界处。

表 138: 适用于共享逻辑的端口（“Shared GT\_COMMON”选项）

名称	方向	宽度
qpll_drp_crscode	输入	12 位
qpll_drp_fsm	输入	18 位
qpll_drp_done	输入	2 位
qpll_drp_reset	输入	2 位
qpll_qplllock	输入	2 位
qpll_qplloutclk	输入	2 位
qpll_qplloutrefclk	输入	2 位
qpll_qplld	输出	1 位
qpll_qpllreset	输出	2 位
qpll_drp_clk	输出	1 位
qpll_drp_rst_n	输出	1 位
qpll_drp_ovrd	输出	1 位
qpll_drp_gen3	输出	1 位
qpll_drp_start	输出	1 位

在 7 系列 Gen2 器件的“Shared Logic”选项卡下选中“Shared Clocking”选项时，下表中的端口就会出现在边界处。

表 139: 适用于共享逻辑的端口（“Shared Clocking”选项）

名称	方向	宽度
pipe_pclk_in	输入	1 位
pipe_rxusrclk_in	输入	1 位
pipe_rxoutclk_in	输入	1 位
pipe_dclk_in	输入	1 位
pipe_userclk1_in	输入	1 位
pipe_userclk2_in	输入	1 位
pipe_oobclk_in	输入	1 位
pipe_mmcm_lock_in	输入	1 位
pipe_mmcm_rst_n	输入	1 位
pipe_txoutclk_out	输出	1 位
pipe_rxoutclk_out	输出	1 位

表 139: 适用于共享逻辑的端口（“Shared Clocking”选项）（续）

名称	方向	宽度
pipe_pclk_sel_out	输出	1 位
pipe_gen3_out	输出	1 位

下表显示了该版本的 IP 中新增的端口。当启用 MSI-X 功能并且器件类型为“PCIe Endpoint”时，此端口在边界处可用。

表 140: 新增端口

名称	方向	宽度
msix_en	输出	1 位

# 调试

本附录包含有关赛灵思支持网站和调试工具上可用资源的详细信息。

## 在 Xilinx.com 上寻求帮助

为了能在使用子系统时帮助设计和调试进程，[赛灵思支持网页](#)上提供了大量关键资源，如产品文档、版本说明、答复记录、已知问题相关信息以及如何获取进一步产品支持的链接。[赛灵思社区论坛](#)还可供会员学习、参与、分享和提出与赛灵思解决方案相关的问题。

### 文档

本产品指南是与该子系统相关的主要文件。本指南以及有助于设计进程的所有产品相关文档都可以在[赛灵思支持网页](#)上找到，也可以通过赛灵思 Documentation Navigator 来获取。您可通过[下载页面](#)下载赛灵思 Documentation Navigator。如需了解此工具和可用功能的详细信息，请在安装后打开联机帮助。

### 解决方案中心

如需了解设计周期各阶段有关器件、软件工具和 IP 等的技术支持，请参阅[赛灵思解决方案中心](#)。相关专题包括设计辅助、建议和故障排除提示等。

请参阅[赛灵思 PCI Express 解决方案中心](#)。

### 答复记录

答复记录包括有关常见问题的信息、有关如何解决这些问题的实用信息以及有关赛灵思产品的所有已知问题。我们每天都会创建和维护答复记录，确保用户可以获取最准确的信息。

您可以通过[赛灵思支持网页](#)（主页）上的“搜索支持”框找到此子系统的答复记录。要最大程度扩展搜索结果范围，请使用适当的关键词，例如：

- 产品名称
- 工具消息
- 所遇到问题的摘要

返回结果后，可以使用过滤器搜索来进一步定位结果。

### DMA/Bridge Subsystem for PCIe 的主答复记录

[答复记录 65443](#)



## 技术支持

赛灵思在[赛灵思社区论坛](#)上为此 LogiCORE™ IP 产品提供技术支持，前提是用户按产品文档中所述方式使用该产品。如果您执行以下任何操作，则赛灵思无法保证产品时序和功能的正常运行，也无法保证提供相应支持：

- 在文档中未定义的器件中实现解决方案。
- 超出产品文档中允许的范围自定义解决方案。
- 更改设计中任何标记有“DO NOT MODIFY”的部分。

如需提问，请导航至[赛灵思社区论坛](#)。

---

## 调试工具

有许多工具可用于解决 DMA/Bridge Subsystem for PCIe 设计问题。至关重要的是要了解哪些工具可用于调试各种情况。

### Vivado Design Suite 调试功能

Vivado® Design Suite 调试功能可以将逻辑分析器和虚拟 I/O 核直接插入到您的设计中。调试功能还支持您设置触发条件，以便在硬件中捕获应用和集成块端口信号。随后，您便可对捕获的信号进行分析。Vivado IDE 中的这个功能用来对在赛灵思器件中运行的设计进行逻辑调试和验证。

Vivado 逻辑分析器用于与下列逻辑调试 LogiCORE IP 核交互：

- ILA 2.0（及更高版本）
- VIO 2.0（及更高版本）

请参阅《Vivado Design Suite 用户指南：编程和调试》([UG908](#))。

## 参考板

各种赛灵思开发板均支持 DMA/Bridge Subsystem for PCIe 核。这些开发板可用于进行原型设计并确定核可与系统通信。

- 7 系列 FPGA 评估板
  - VC709
  - KC705
- UltraScale™ FPGA 评估板
  - KCU105
  - VCU108
- UltraScale+™
  - KCU116
  - VCU118

- ZCU106

## 硬件调试

硬件问题各不相同，可能是链路初始化问题，也可能是测试数小时后才能注意到的问题。本节提供了常见问题的调试步骤。Vivado® 调试功能是可用于硬件调试的宝贵资源。可以通过调试功能来探测以下各个部分中提到的信号名称，以便对特定问题进行调试。

### 常规检查

确保核的所有时序约束都已正确整合在设计示例中，并且在实现期间已满足所有约束条件。

- 在布局布线后的时序仿真中是否能够正常工作？如果在硬件中发现问题，但在时序仿真中没有出现问题，则可能表示存在 PCB 问题。确保所有时钟源均处于活动状态且无任何错误。
- 如果在设计中使用 MMCM，请通过监控 `locked` 端口确保所有 MMCM 都被锁定。
- 如果您的输出为 0，请检查您的许可。

### DMA/Bridge Subsystem for PCIe 的初始调试

位于每个引擎范围之外的状态位可用于对子系统进行初始调试。每个通道接口均可为用户应用提供重要的状态。

表 141: 子系统的初始调试

位索引	字段	描述
6	Run	通道控制寄存器运行位。
5	IRQ_Pending	当通道包含暂挂中断时，该位断言有效。
4	Packet_Done	在 AXIST 接口上，该位表示 EOP 位所指示的最后一项数据已转发。
3	Descriptor_Done	描述符已完成将数据从源转发到目标的传输操作。
2	Descriptor_Stop	描述符中的 Descriptor_Done and Stop 位已置位。
1	Descriptor_Completed	描述符中的 Descriptor_Done and Completed 位已置位。
0	Busy	通道描述符缓冲器不为空，或者 DMA 请求尚未完成。

#### 相关信息

[通道 0-3 状态端口](#)

# 使用赛灵思虚拟线缆进行调试

赛灵思虚拟电缆 (XVC) 允许 Vivado® Design Suite 通过非 JTAG 接口连接到 FPGA 调试核。标准 Vivado® Design Suite 调试功能使用 JTAG 连接到物理硬件 FPGA 资源并通过 Vivado 执行调试。本章节主要聚焦如何使用 XVC 通过 PCIe® 链路而不是标准 JTAG 调试接口来执行调试。此过程称为 XVC-over-PCIe，支持 Vivado ILA 波形捕获、VIO 调试控制以及使用 PCIe 链路作为信道与其它赛灵思调试核进行交互。

当 JTAG 调试不可用时，XVC-over-PCIe 应用于使用 Vivado Design Suite 调试功能来远程执行 FPGA 调试。此方法常用于数据中心应用，在此类应用中，FPGA 连接到 PCIe 主机系统，且不连接到任何其它硬件器件。

使用通过 XVC 进行调试的方法需要软件、驱动和 FPGA 硬件设计组件。由于 XVC-over-PCIe 调试涉及 FPGA 硬件设计组件，因此只有在 FPGA 已完成加载且具备 FPGA 硬件设计用于实现 XVC-over-PCIe 并且已建立到主机 PC 的 PCIe 链路之后，才能执行调试。通常要完成这些操作，需先将启用 XVC-over-PCIe 的设计加载到板上的配置闪存中，然后再将卡插入数据中心位置。由于使用 XVC-over-PCIe 进行调试与 PCIe 信道有关，因此，此方法不应用于调试 PCIe 链路相关问题。



**重要提示！** XVC 仅提供到 FPGA 内部调试核的连接。它不提供器件编程或访问器件 JTAG 和配置寄存器的功能。这些操作可通过其它标准赛灵思接口或外设（如 PCIe MCAP VSEC 和 HWICAP IP）来执行。

## 简介

支持 XVC-over-PCIe 调试的主要组件包括：

- 主机 PC XVC-Server 应用
- 主机 PC PCIe-XVC 驱动
- 启用 XVC-over-PCIe 的 FPGA 设计

这些组件可作为参考，以供演示如何为赛灵思 FPGA 设计创建 XVC 连接。这 3 个组件如下图所示，并通过 TCP/IP 套接字连接到 Vivado Design Suite 调试特性。

图 34：XVC-over-PCIe 软件和硬件组件



X18837-101.420

## 主机 PC XVC-Server 应用

使用调试功能时，`hw_server` 应用由 Vivado Design Suite 启动。您可通过 Vivado IDE 将 `hw_server` 连接到本地或远程 FPGA 目标。此接口同样可用于连接到本地或远程 PCIe-XVC 目标。主机 PCIe XVC-Server 应用使用 TCP/IP 套接字连接到赛灵思 `hw_server`。这样即可允许 Vivado（使用 `hw_server`）和 XVC-Server 应用在同一台 PC 上运行或者在通过以太网连接的不同 PC 上运行。XVC-Server 应用需在直接连接到 FPGA 硬件资源的 PC 上运行。在此情况下，FPGA 硬件通过 PCIe® 连接到主机 PC。XVC-Server 应用通过同样在主机 PC 上运行的 PCIe-XVC 驱动来连接到 FPGA 硬件器件。

## 主机 PC XVC-over-PCIe 驱动

XVC-over-PCIe 驱动可提供与连接到主机 PC 并启用 PCIe 的 FPGA 硬件资源的连接。因此，此驱动作为 Linux 内核模式驱动提供，用于访问位于以下位置的 PCIe 硬件器件：<Vivado\_Installation\_Path>/data/xicom/driver/pcie/xvc\_pcie.zip。此驱动的必要组件必须添加到为特定 FPGA 平台创建的驱动中。此驱动用于实现 XVC-Server 应用通过 PCIe 与 FPGA 进行通信所需的基本功能。

## 启用 XVC-over-PCIe 的 FPGA 设计

传统上，Vivado® 调试是通过 JTAG 来执行的。默认情况下，Vivado 工具自动化可将赛灵思调试核连接到 FPGA 中的 JTAG BSCAN 资源以执行调试。为执行 XVC-over-PCIe 调试，此信息必须通过 PCIe 链路而不是 JTAG 电缆接口来发射。赛灵思 Debug Bridge IP 支持您将调试网络通过 PCIe 扩展配置接口 (PCIe-XVC-VSEC) 或通过 PCIe BAR 的 AXI4-Lite 内存映射接口 (AXI-XVC) 连接到 PCIe。

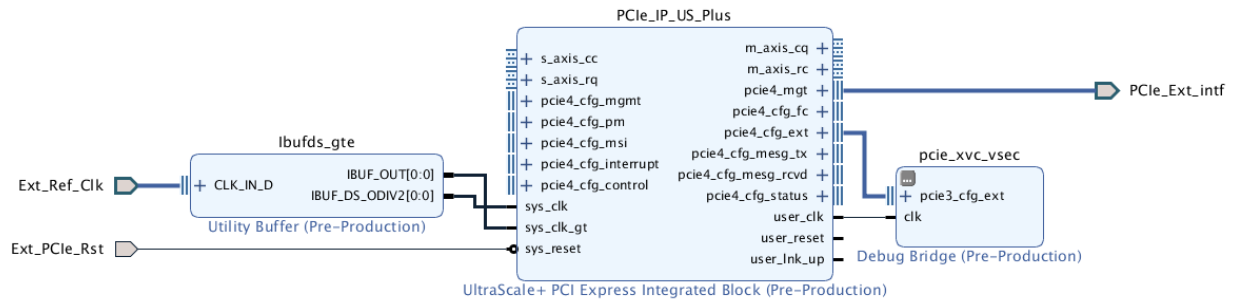
Debug Bridge IP 配置为 “From PCIe to BSCAN” 或 “From AXI to BSCAN” 之后即可为分别源于 PCIe 扩展功能或 AXI4-Lite 接口的赛灵思调试网络提供连接点。Vivado 工具自动化可将此 Debug Bridge 实例连接到设计中的赛灵思调试核，而不是将其连接到 JTAG BSCAN 接口。在判定将 Debug Bridge 连接到 PCIe 扩展配置空间还是 AXI4-Lite 之间，设计需权衡利弊。以下章节描述了这 2 种实现的实现注意事项和寄存器映射。

### 通过 PCIe 扩展配置空间的 XVC-over-PCIe (PCIe-XVC-VSEC)

通过使用 PCIe-XVC-VSEC 方法，Debug Bridge IP 即可使用 PCIe 供应商限定的扩展功能 (VSEC) 来实现从 PCIe 到 Debug Bridge IP 的连接。PCIe 扩展配置空间设置为可从主机 PC 交付的扩展功能的链接列表。这对于部分平台尤为实用，在此类平台中仅限通过某一版本的设计来实现 PCIe-XVC-VSEC，其它设计实现则无效。此链接列表可用于检测 PCIe-XVC-VSEC 是否存在并予以相应的响应。

PCIe 扩展配置接口使用 PCIe 配置传输事务，而不是 PCIe 内存 BAR 传输事务。虽然 PCIe 配置传输事务慢得多，但这些事务不会在 PCIe IP 边界处与 PCIe 内存 BAR 传输事务相互干涉。这样即可在 FPGA 中建立独立的数据和调试通信路径。如果您预计将对数据路径进行调试，那么这是理想的方法。即使数据路径受损或中断，PCIe 扩展配置接口仍可保持运行以执行调试。下图描述了 PCIe IP 与 Debug Bridge IP 之间用于实现 PCIe-XVC-VSEC 的连接。

图 35: 含 PCIe 扩展功能接口的 XVC-over-PCIe

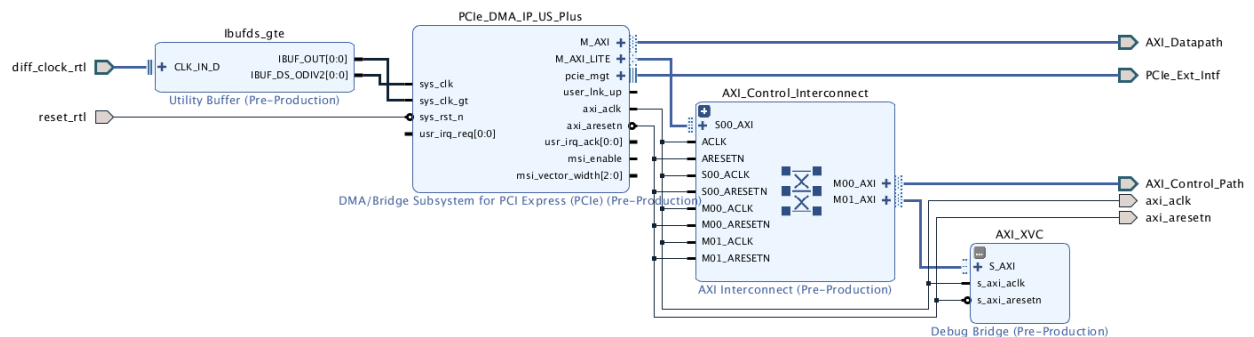


**注释:** 虽然上图仅显示 UltraScale+™ 器件 Integrated Block for PCIe IP, 但其它 PCIe IP (即, UltraScale™ 器件 Integrated Block for PCIe、AXI Bridge for PCIe 或 PCIe DMA IP) 均可在此图中互换使用。

## 通过 AXI 的 XVC-over-PCIe (AXI-XVC)

通过使用 AXI-XVC 方法, Debug Bridge IP 即可通过 AXI Interconnect IP 连接至 PCIe IP。Debug Bridge IP 采用与其它 AXI4-Lite 从接口 IP 相似的方法连接至 AXI Interconnect, 并且同样要求为其分配特定的地址范围。原先此配置中的 debug\_bridge IP 连接到控制路径网络, 而不是系统数据路径网络。下图描述了 DMA Subsystem for PCIe IP 与 Debug Bridge IP 之间用于此实现的连接。

图 36: 含 AXI4-Lite 接口的 XVC over PCIe



**注释:** 虽然上图显示了 PCIe DMA IP, 但在此图中, 任何支持 AXI 的 PCIe IP 均可互换使用。

AXI-XVC 实现支持更高速的传输事务。但 XVC 调试流量与其它 PCIe 控制路径流量会通过相同的 PCIe 端口进行传递和互连, 导致在此路径上执行传输事务调试的难度增加。因此, 应使用 AXI-XVC 调试来调试特定的外设或差分 AXI 网络, 而不是尝试对与 AXI-XVC 调试通信路径重叠的数据路径进行调试。

## XVC-over-PCIe 寄存器映射

PCIe-XVC-VSEC 和 AXI-XVC 包含的寄存器映射稍有不同, 在设计 XVC 驱动和软件时必须考量其差异。下表中的寄存器映射显示了对应基址的字节偏移。

- PCIe-XVC-VSEC 基址必须位于 PCIe 扩展配置空间的有效范围内。此范围在 Debug Bridge IP 配置内指定。
- AXI-XVC Debug Bridge 的基址是 Debug Bridge IP 的偏移, 此偏移在 Vivado 地址编辑器中指定。

下表描述了针对“From PCIe-Ext to BSCAN”模式或“From AXI to BSCAN”模式进行配置时的 Debug Bridge IP 的寄存器映射（作为对应基址的偏移）。

表 142: Debug Bridge 对应 XVC-PCIe-VSEC 寄存器映射

寄存器偏移	寄存器名称	描述	寄存器类型
0x00	PCIe Ext Capability Header	PCIe 定义的供 VSEC 使用的字段。	只读
0x04	PCIe VSEC Header	PCIe 定义的供 VSEC 使用的字段。	只读
0x08	XVC Version Register	IP 版本和功能信息。	只读
0x0C	XVC Shift Length Register	移位长度。	读写
0x10	XVC TMS Register	TMS 数据。	读写
0x14	XVC TDIO Register	TDO/TDI 数据。	读写
0x18	XVC Control Register	通用控制寄存器。	读写
0x1C	XVC Status Register	通用状态寄存器。	只读

表 143: Debug Bridge 对应 AXI-XVC 寄存器映射

寄存器偏移	寄存器名称	描述	寄存器类型
0x00	XVC Shift Length Register	移位长度。	读写
0x04	XVC TMS Register	TMS 数据。	读写
0x08	XVC TDI Register	TDI 数据。	读写
0x0C	XVC TDO Register	TDO 数据。	只读
0x10	XVC Control Register	通用控制寄存器。	读写
0x14	XVC Status Register	通用状态寄存器。	只读
0x18	XVC Version Register	IP 版本和功能信息。	只读

## PCIe Ext Capability Header

此寄存器用于识别添加到 PCIe 设计的 PCIe-XVC-VSEC。PCIe Ext Capability Header 中的字段和值由 PCI-SIG 定义，用于识别扩展功能的格式，并提供指向下一项扩展功能（如果适用）的指针。当用作 PCIe-XVC-VSEC 时，应先对相应的 PCIe ID 字段求值，然后再进行解读。这些字段包括 PCIe 供应商 ID、PCIe 器件 ID、PCIe 版本 ID、子系统供应商 ID 和子系统 ID。提供的驱动会专门检查 PCIe 供应商 ID 与赛灵思 ID (0x10EE) 是否匹配，然后再解读此寄存器。下表描述了此寄存器中的字段。

表 144: PCIe Ext Capability Header 寄存器描述

位的位置	字段	描述	初始值	类型 (Type)
15:0	PCIe Extended Capability ID	该字段为 PCI-SIG 定义的 ID 号，表示扩展功能 (Extended Capability) 的性质和格式。VSEC 的 Extended Capability ID 为 0x000B	0x000B	只读
19:16	Capability Version	该字段为 PCI-SIG 定义的版本号，表示存在的功能结构的版本。针对此版本的规格，该编号必须为 0x1。	0x1	只读
31:20	Next Capability Offset	该字段从用户传入，包含距离下一个 PCI Express 功能结构的偏移，或者如果已链接的功能列表没有任何其它项，则该字段为 0x000。对于 PCIe 扩展配置空间内实现的扩展功能，该值必须始终保持在 PCIe 扩展配置空间的有效范围内。	0x000	只读

## PCIe VSEC 报头（仅限 PCIe-XVC-VSEC）

当 Debug Bridge IP 位于此模式下时，此寄存器用于识别 PCIe-XVC-VSEC。这些字段由 PCI-SIG 定义，但其值则取决于供应商 ID（赛灵思 ID 为 0x10EE）。应先限定 PCIe 扩展功能报头寄存器值，然后再解读此寄存器。

表 145: PCIe XVC VSEC 报头寄存器描述

位的位置	字段	描述	初始值	类型 (Type)
15:0	VSEC ID	该字段所表示的 ID 值可用于识别 PCIe-XVC-VSEC，其值取决于供应商 ID（赛灵思 ID 为 0x10EE）。	0x0008	只读
19:16	VSEC 版本	该字段所表示的版本 ID 值可用于识别 PCIe-XVC-VSEC 版本。	0x0	只读
31:20	VSEC 长度	该字段表示整个 PCIe-XVC-VSEC 结构中的字节数，包括 PCIe 扩展功能报头寄存器和 PCIe VSEC 报头寄存器。	0x020	只读

## XVC Version Register（仅限 PCIe-XVC-VSEC）

此寄存器是由赛灵思工具填充的，供 Vivado Design Suite 用于识别硬件设计中实现的 Debug Bridge IP 的特定功能。

## XVC Shift Length Register

此寄存器用于在调试扫描链内设置扫描链移位长度。

## XVC TMS Register

此寄存器用于在调试扫描链中设置 TMS 数据。

## XVC TDO/TDI 数据寄存器

此寄存器用于 TDO/TDI 数据访问。使用 PCIePCI-XVC-VSEC 时，这 2 个寄存器组合到单一字段中。使用 AXI-XVC 时，则作为 2 个独立寄存器来实现。

## XVC Control Register

此寄存器用于 XVC 控制数据。

## XVC Status Register

此寄存器用于 XVC 状态信息。

## XVC 驱动和软件

在 Vivado Design Suite 安装过程中提供了 XVC 驱动和软件示例，位置如下：<Vivado\_Installation\_Path>/data/xicom/driver/pcie/xvc-pcie.zip。在将 XVC 功能集成到赛灵思 FPGA 平台设计驱动和软件时，应使用此示例作为参考。所提供的 Linux 内核模式驱动和软件可用于为 PCIe-XVC-VSEC 实现和 AXI-XVC Debug Bridge 实现执行 XVC-over-PCIe 调试。



在 PCIe-XVC-VSEC 模式下工作时，此驱动将发起 PCIe 配置传输事务，以便与 FPGA 调试网络相连。在 AXI-XVC 模式下工作时，此驱动将发起 32 位 PCIe 内存 BAR 传输事务，以便与 FPGA 调试网络相连。默认情况下，此驱动将尝试发现 PCIe-XVC-VSEC，如果在已链接的 PCIe 配置扩展功能列表中未找到 PCIe-XVC-VSEC，则使用 AXI-XVC。

所提供的驱动位于 Vivado 安装数据目录中，采用 .zip 文件格式。此 .zip 文件应复制到通过 PCIe 连接至赛灵思 FPGA 的主机 PC 上，并解压以供使用。其中包含 README.txt 文件；请复查这些文件，以获取有关安装和运行 XVC 驱动和软件的说明。

## 有关串联配置设计或 Dynamic Function eXchange 设计的特殊注意事项

串联配置和 Dynamic Function eXchange (DFX) 设计可能具有额外的注意事项需要考量，因为这些流程会将物理资源分区到多个独立区域内。在将调试 IP 添加到设计中（例如，VIO、ILA、MDM 和 MIG-IP）时，应考量使用这些物理分区。专为“From PCIe-ext to BSCAN”或“From AXI to BSCAN”配置的 Debug Bridge IP 只能放置在设计的静态分区内。在 DFX 或串联现场更新区域内使用调试 IP 时，应将另一个调试 BSCAN 接口添加到动态区域模块定义中，并在动态区域模块例化过程中使其保持未连接状态。

要将 BSCAN 接口添加到可重配置分区定义中，应将相应的端口和端口属性添加到可重配置分区定义中。以下提供的 Verilog 可作为将 BSCAN 接口添加到端口声明的模板。

```
...
// BSCAN interface definition and attributes.
// This interface should be added to the DFX module definition
// and left unconnected in the DFX module instantiation.
(* X_INTERFACE_INFO = "xilinx.com:interface:bscan:1.0 S_BSCAN drck" *)
(* DEBUG="true" *)
input S_BSCAN_drck,
(* X_INTERFACE_INFO = "xilinx.com:interface:bscan:1.0 S_BSCAN shift" *)
(* DEBUG="true" *)
input S_BSCAN_shift,
(* X_INTERFACE_INFO = "xilinx.com:interface:bscan:1.0 S_BSCAN tdi" *)
(* DEBUG="true" *)
input S_BSCAN_tdi,
(* X_INTERFACE_INFO = "xilinx.com:interface:bscan:1.0 S_BSCAN update" *)
(* DEBUG="true" *)
input S_BSCAN_update,
(* X_INTERFACE_INFO = "xilinx.com:interface:bscan:1.0 S_BSCAN sel" *)
(* DEBUG="true" *)
input S_BSCAN_sel,
(* X_INTERFACE_INFO = "xilinx.com:interface:bscan:1.0 S_BSCAN tdo" *)
(* DEBUG="true" *)
output S_BSCAN_tdo,
(* X_INTERFACE_INFO = "xilinx.com:interface:bscan:1.0 S_BSCAN tms" *)
(* DEBUG="true" *)
input S_BSCAN_tms,
(* X_INTERFACE_INFO = "xilinx.com:interface:bscan:1.0 S_BSCAN tck" *)
(* DEBUG="true" *)
input S_BSCAN_tck,
(* X_INTERFACE_INFO = "xilinx.com:interface:bscan:1.0 S_BSCAN runtest" *)
(* DEBUG="true" *)
input S_BSCAN_runtest,
(* X_INTERFACE_INFO = "xilinx.com:interface:bscan:1.0 S_BSCAN reset" *)
(* DEBUG="true" *)
input S_BSCAN_reset,
(* X_INTERFACE_INFO = "xilinx.com:interface:bscan:1.0 S_BSCAN capture" *)
(* DEBUG="true" *)
```



```
input S_BSCAN_capture,
(* X_INTERFACE_INFO = "xilinx.com:interface:bscan:1.0 S_BSCAN bscanid_en" *)
(* DEBUG="true" *)
input S_BSCAN_bscanid_en,
....
```

运行 `link_design` 时，公开的端口将通过工具自动化连接到调试网络的静态部分。ILA 也会根据设计需要连接到调试网络。在设计顶层可能还会添加另一个 `dbg_hub` 单元。对于含现场更新的串联设计，`dbg_hub` 和工具插入的时钟缓存必须添加到相应的设计分区。以下是完成 `opt_design` 后可运行的 Tcl 命令示例，这些命令用于将 `dbg_hub` 原语与相应的设计分区加以关联。

```
# Add the inserted dbg_hub cell to the appropriate design partition.
set_property HD.TANDEM_IP_PBLOCK Stage1_Main [get_cells dbg_hub]
# Add the clock buffer to the appropriate design partition.
set_property HD.TANDEM_IP_PBLOCK Stage1_Config_IO [get_cells
dma_pcie_0_support_i/
pcie_ext_cap_i/vsec_xvc_inst/vsec_xvc_dbg_bridge_inst/inst/bsip/ins
t/USE_SOFTBSCAN.U_TAP_TCKBUFG]
```

## 使用 PCIe-XVC-VSEC 设计示例

PCIe-XVC-VSEC 已集成到 PCIe 设计示例中，包含在 UltraScale+™ Integrated Block for PCIe IP 的“高级 (Advanced)”设置内。本章节旨在提供相关指示信息，以指导您如何使用 PCIe-XVC-VSEC 生成 PCIe 设计示例，然后使用所提供的 XVC 驱动和软件通过 PCIe 对 FPGA 进行调试。此示例是在客户应用中使用 XVC 的示例。FPGA 设计、驱动和软件元素都需要集成到客户设计中。

### 生成 PCIe-XVC-VSEC 设计示例

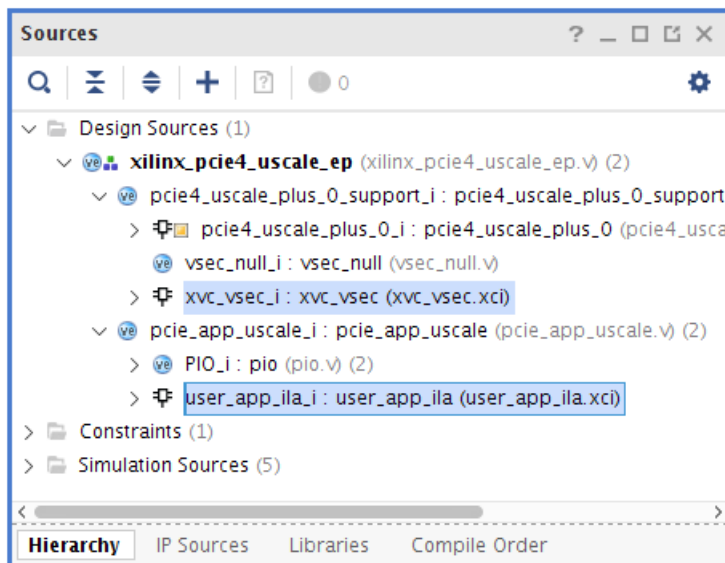
通过选择以下选项，可将 PCIe-XVC-VSEC 添加到 UltraScale+™ PCIe 设计示例。

1. 将核配置为期望的配置。
2. 在“基本信息 (Basic)”选项卡上，选中“Advanced”模式。
3. 在“高级选项 3 (Adv. Options-3)”选项卡上：
  - a. 选中“PCI Express Extended Configuration Space Enable”复选框，以启用 PCI Express 扩展配置接口。这样即可将附加扩展功能添加到 PCI Express 核中。
  - b. 选中“Add the PCIe-XVC-VSEC to the Example Design”复选框即可在设计示例生成中启用 PCIe-XVC-VSEC。
4. 验证 PCIe IP 的其它配置选项。以下选项是配置驱动程序用于硬件实现所必需的选项：
  - PCIe Vendor ID (0x10EE 对应赛灵思)
  - PCIe Device ID (取决于用户选择)
5. 单击“OK”以最终确认选择，并生成 IP。
6. 为应用所需的 IP 生成输出文件。
7. 在“Sources”窗口中，右键单击 IP 并选择“Open IP Example Design”。
8. 选择用于生成设计示例的目录，然后单击“OK”。

生成后的设计示例会显示：

- PCIe IP 已连接到支持封装器中的 `xvc_vsec`，并且
- 已向设计的用户应用部分添加了 ILA IP。

此处显示了 FPGA 设计的硬件部分所需的连接。可根据您的应用的实际需求来添加其它调试核。



**注释：**虽然上图所示仅适用于 UltraScale+ 器件 Integrated Block for PCIe IP，但对于其它 PCIe IP 而言，设计示例层级是相同的。

9. 双击识别为 `xvc_vsec` 的 Debug Bridge IP 即可查看此 IP 的配置选项。记录以下配置参数，因为这些参数将用于配置驱动程序。
  - PCIe XVC VSEC ID（默认值为 `0x0008`）
  - PCIe XVC VSEC Rev ID（默认值为 `0x0`）



**重要提示！**使用赛灵思供应商 ID 或提供的 XVC 驱动程序和软件时，请勿修改这些参数值。这些值用于检测 XVC 扩展功能。（请参阅 PCIe 规范，以获取其它详细信息。）

10. 在 Flow Navigator 中，单击“Generate Bitstream”即可为设计示例工程生成比特流。随后，此比特流将被加载到 FPGA 板上，以启用基于 PCIe 的 XVC 调试。

完成基于 PCIe 的 XVC (XVC-over-PCIe) 硬件设计后，可使用已启用 XVC 的相应 PCIe 驱动程序和关联的 XVC-Server 软件应用来将 Vivado Design Suite 连接到 PCIe 连接的 FPGA。Vivado 可连接到本地机器上运行的 XVC-Server 应用，或者也可以使用 TCP/IP 套接字远程连接到其它机器上的应用。

## 系统初始化

第一步是对系统上的 FPGA 和功耗进行编程，使主机系统可检测到 PCIe 链路。这可通过下列任一方式完成：

- 将设计文件编程到 FPGA 板上存在的闪存中，或者
- 直接通过 JTAG 对器件进行编程。

如果卡由主机 PC 供电，则需要将其上电后才能使用 JTAG 来执行编程，然后重新启动以允许 PCIe 链路执行枚举。系统启动并正常运行后，您可使用 Linux `lspci` 实用工具来列出基于 FPGA 的 PCIe 器件的详细信息。

## 编译和加载驱动

所提供的 PCIe 驱动和软件应根据特定平台进行自定义。要完成此操作，所开发的驱动和软件通常会先验证供应商 ID、器件 ID、版本 ID、子系统供应商 ID 和子系统 ID，然后再尝试访问器件扩展功能或外设（如 PCIe-XVC-VSEC 或 AXI-XVC）。由于提供的驱动为通用型驱动，因此它仅验证供应商 ID 和器件 ID 以确认兼容性，然后再尝试验证 PCIe-XVC-VSEC 或 AXI-XVC 外设。

XVC 驱动和软件均作为 ZIP 文件提供，包含在 Vivado Design Suite 安装内。

1. 请将此 ZIP 文件从 Vivado 安装目录复制到 FPGA 连接的主机 PC 并解压其内容。此文件位于 Vivado 安装目录内的以下路径中。

XVC Driver and SW Path: .../data/xicom/driver/pcie/xvc\_pcie.zip

driver\_\* 和 xvcserver 目录内的 README.txt 文件用于识别如何编译、安装和运行 XVC 驱动和软件，详细步骤汇总如下。将驱动和软件文件复制到主机 PC 并以具有 root 用户权限的用户身份登录后，请遵循以下步骤进行操作。

2. 修改 driver\_\*/xvc\_pcie\_user\_config.h 文件中的变量以匹配您的硬件设计和 IP 设置。请考虑修改以下变量：
  - PCIE\_VENDOR\_ID: PCIe® IP 自定义中定义的 PCIe 供应商 ID。
  - PCIE\_DEVICE\_ID: PCIe® IP 自定义中定义的 PCIe 器件 ID。
  - Config\_space: 允许选择使用 PCIe-XVC-VSEC 或 AXI-XVC 外设。默认值 AUTO 会首先尝试发现 PCIe-XVC-VSEC，然后，如果未找到 PCIe-XVC-VSEC，则会尝试连接至 AXI-XVC 外设。值 CONFIG 或 BAR 可用于在 PCIe-XVC-VSEC 实现和 AXI-XVC 实现之间明确选择所期望的实现。
  - config\_vsec\_id: 当“桥接类型 (Bridge Type)”配置为“从 PCIe 到 BSCAN (From PCIe to BSCAN)”时，该值为在“Debug Bridge IP”中定义的 PCIe XVC VSEC ID（默认值 0x0008）。该值仅用于检测 PCIe-XVC-VSEC。
  - config\_vsec\_rev: 当“桥接类型 (Bridge Type)”配置为“从 PCIe 到 BSCAN (From PCIe to BSCAN)”时，该值为在“Debug Bridge IP”中定义的 PCIe XVC VSEC Rev ID（默认值 0x0）。该值仅用于检测 PCIe-XVC-VSEC。
  - bar\_index: PCIe BAR 索引，当“桥接类型”配置为“从 AXI 到 BSCAN (From AXI to BSCAN)”时，应使用此索引值来访问“Debug Bridge IP”。此 BAR 索引指定为 PCIe IP 自定义与系统设计中可寻址的 AXI 外设的组合。该值仅用于检测 AXI-XVC 外设。
  - bar\_offset: PCIe BAR 偏移，当“桥接类型”配置为“从 AXI 到 BSCAN (From AXI to BSCAN)”时，应使用此偏移值来访问“Debug Bridge IP”。此 BAR 偏移指定为 PCIe IP 自定义与系统设计中可寻址的 AXI 外设的组合。该值仅用于检测 AXI-XVC 外设。
3. 请将源文件移至您所选的目录内。例如，请使用：

```
/home/username/xil_xvc or /usr/local/src/xil_xvc
```

4. 请确保您具有 root 用户权限，并切换至包含驱动文件的目录。

```
# cd /driver_*/
```

5. 编译驱动模块：

```
# make install
```

内核模块对象文件将安装为：

```
/lib/modules/[KERNEL_VERSION]/kernel/drivers/pci/pcie/Xilinx/xil_xvc_driver.ko
```

- 运行 `depmod` 命令以选择新安装的内核模块:

```
# depmod -a
```

- 请确保未加载任何旧版本的驱动:

```
# modprobe -r xil_xvc_driver
```

- 加载模块:

```
# modprobe xil_xvc_driver
```

如果运行 `dmesg` 命令, 您将看到以下消息:

```
kernel: xil_xvc_driver: Starting...
```

**注释:** 您还可在内核对象文件上使用 `insmod` 来加载模块:

```
# insmod xil_xvc_driver.ko
```

但由于与旧内核存在兼容性问题, 因此不推荐此方法, 如非必要请勿使用。

- 生成的字符文件 `/dev/xil_xvc/cfg_ioc0` 归用户根和组根所有, 并且它需要具备 `660` 的权限。如果此文件不允许应用与驱动进行交互, 请更改其中的权限。

```
# chmod 660 /dev/xil_xvc/cfg_ioc0
```

- 为驱动构建简单的测试程序:

```
# make test
```

- 运行测试程序:

```
# ./driver_test/verify_xil_xvc_driver
```

您应可看到各项不同长度的测试成功消息, 后接以下消息:

```
"XVC PCIE Driver Verified Successfully!"
```

## 编译和启动 XVC-Server 应用

XVC-Server 应用可提供 Vivado 硬件服务器与启用 XVC 的 PCIe 器件驱动之间的连接。Vivado Design Suite 使用 TCP/IP 连接至 XVC-Server。所需端口号将需要通过您的网络防火墙正常公开。以下步骤可用于使用默认端口号 `10200` 来编译和启动 XVC 软件应用。

- 请确保系统上的防火墙设置已公开将用于连接到 Vivado Design Suite 的端口。对于此示例, 使用的是端口 `10200`。
- 记录主机名或 IP 地址。需要该主机名和端口号才能将 Vivado 连接到 `xvcserver` 应用。请参阅操作系统帮助页面以获取有关对应您的操作系统的防火墙端口设置的信息。
- 请将源文件移至您所选的目录内。例如, 请使用:

```
/home/username/xil_xvc 或 /usr/local/src/xil_xvc
```

- 切换到包含应用源文件的目录:

```
# cd ./xvcserver/
```

5. 编译该应用:

```
# make
```

6. 启动 XVC-Server 应用:

```
# ./bin/xvc_pcie -s TCP::10200
```

当 Vivado Design Suite 已连接到 XVC-Server 应用后, 在 XVC-Server 中应显示以下消息。

```
Enable verbose by setting VERBOSE env var.
Opening /dev/xil_xvc/cfg_ioc0
```

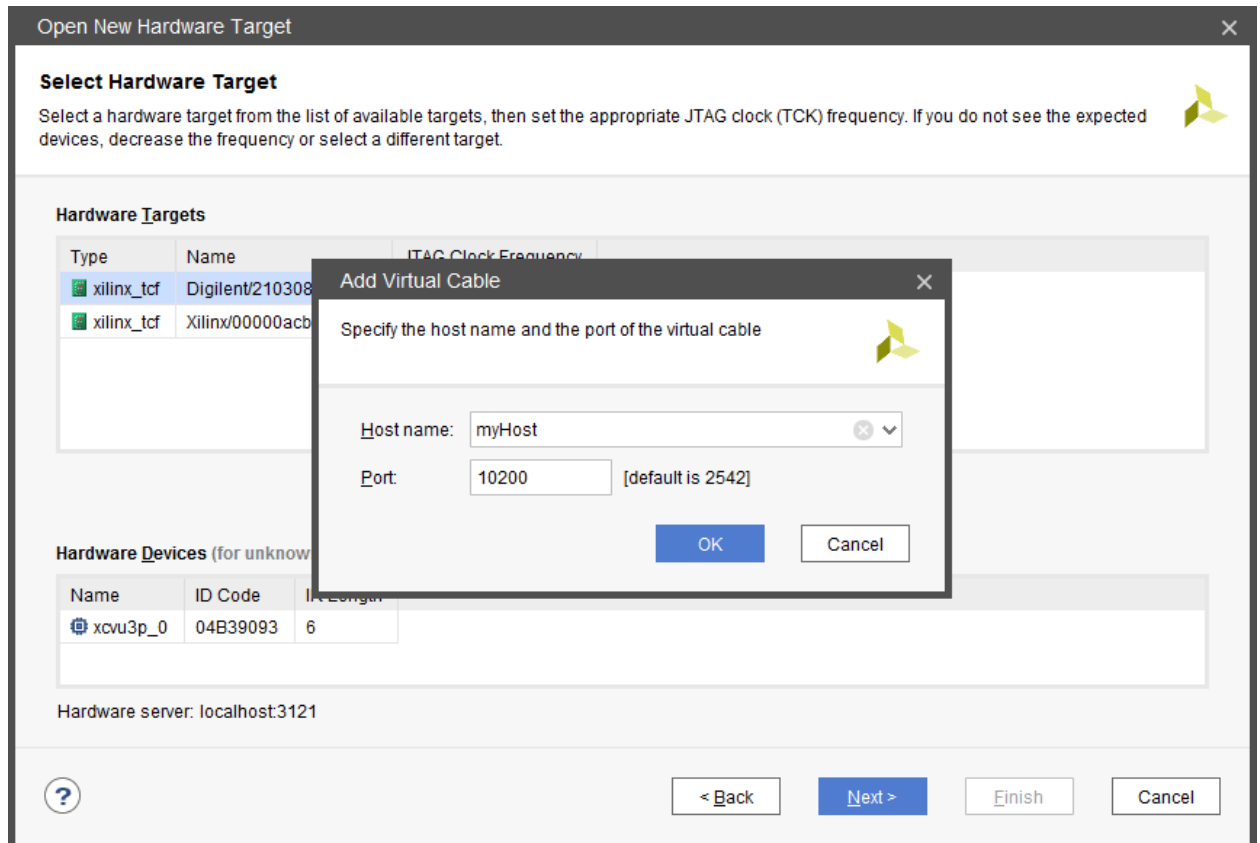
## 将 Vivado Design Suite 连接到 XVC-Server 应用

Vivado Design Suite 可在运行 XVC-Server 应用的计算机上运行, 或者也可以在通过以太网网络连接的另一台计算机上远程运行。但端口必须可供运行 Vivado 的机器访问。要将 Vivado 连接到 XVC-Server 应用, 请遵循以下步骤进行操作, 这些步骤使用默认端口号。

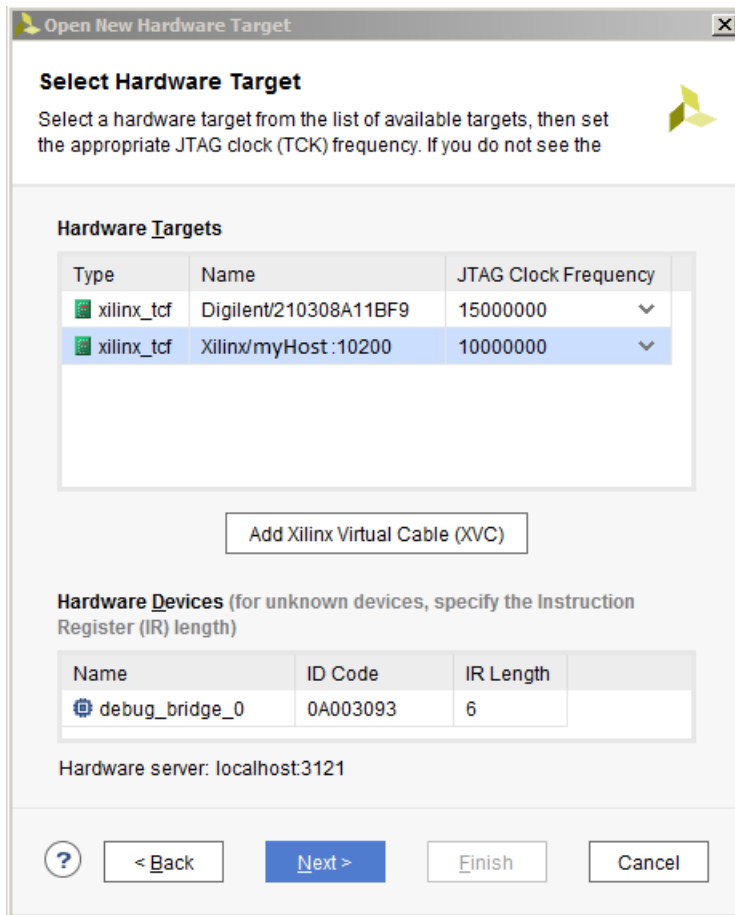
1. 启动 Vivado Design Suite。
2. 选择 “Open HW Manager” 。
3. 在硬件管理器 (Hardware Manager) 中, 依次单击 “Open target” → “Open New Target” 。
4. 单击 “Next” 。
5. 选择 “Local server”, 然后单击 “Next” 。

这样即可在本地机器上启动 `hw_server`, 随后它会连接到 `xvcserver` 应用。

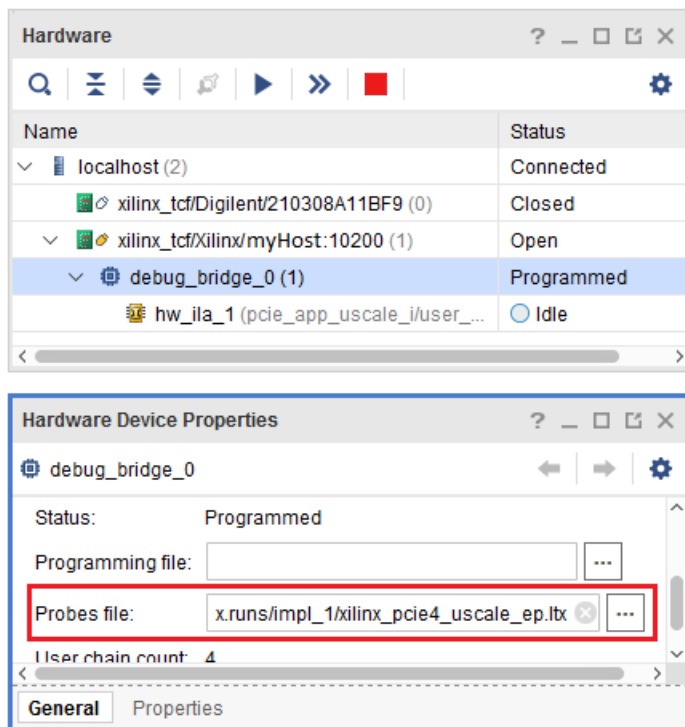
6. 选择 “Add Xilinx Virtual Cable (XVC)” 。
7. 在 “添加虚拟电缆 (Add Virtual Cable)” 对话框中, 输入相应的主机名 (Host name) 或 IP 地址和端口 (Port) 以连接到 `xvcserver` 应用。单击 “OK” 。



- 从“硬件目标 (Hardware Targets)”表中选择新添加的 XVC 目标，然后单击“Next”。



9. 单击“Finish”。
10. 在“硬件器件属性 (Hardware Device Properties)”面板中，选择 Debug Bridge 目标，并指定相应的探测 .ltx 文件。



这样 Vivado 即可识别您的调试核和调试信号，并且您可通过 Vivado 硬件工具接口使用标准调试方法来调试自己的设计。

由此，您即可通过 PCIe 连接而不是通过使用赛灵思虚拟电缆技术的 JTAG 来调试赛灵思 FPGA 设计。您可在 Vivado 中使用右键单击菜单来关闭硬件服务器，以终止连接。如果 PCIe 连接中断或者 XVC-Server 应用停止运行，则与 FPGA 和关联的调试核的连接也将中断。

## 运行时间注意事项

对器件进行编程时，不应运行 Vivado 到 XVC-Server 应用的连接。XVC-Server 及其与 Vivado 的连接应仅限在器件完成编程且硬件 PCIe 接口处于活动状态时才能启动。

对于 DFX 设计，在 DFX 操作期间终止连接至关重要。在 DFX 操作期间，如果动态区域内部存在调试核，那么应对部分调试树进行重新编程。DFX 操作期间，Vivado 调试工具不应通过 XVC 与 FPGA 主动进行通信。



## 附加资源与法律提示

### 赛灵思资源

如需获取答复记录、技术文档、下载以及论坛等支持性资源，请参阅[赛灵思技术支持](#)。

### Documentation Navigator 与设计中心

赛灵思 Documentation Navigator (DocNav) 提供了访问赛灵思文档、视频和支持资源的渠道，您可以在其中筛选搜索信息。打开 DocNav 的方法：

- 在 Vivado® IDE 中，选择 “Help” → “Documentation and Tutorials”。
- 在 Windows 上，选择 “Start” → “All Programs” → “Xilinx Design Tools” → “DocNav”。
- 在 Linux 命令提示中输入 `docnavo`。

赛灵思设计中心提供了根据设计任务和其它主题整理的文档链接，可供您用于了解关键概念以及常见问题解答。要访问设计中心，请执行以下操作：

- 在 DocNav 中，单击 “Design Hubs View” 选项卡。
- 在赛灵思网站上，查看[设计中心](#)页面。

**注释：**如需了解有关 DocNav 的更多信息，请参阅赛灵思网站上的 [Documentation Navigator](#)。

### 参考资料

以下技术文档是非常实用的补充资料，可配合本指南一起使用：

1. 《AMBA AXI4-Stream 协议规范》([ARM IHI 0051A](#))
2. PCI-SIG 文档 ([www.pcisig.com/specifications](http://www.pcisig.com/specifications))
3. 《Vivado Design Suite: AXI 参考指南》([UG1037](#))
4. 《AXI Bridge for PCI Express Gen3 Subsystem 产品指南》([PG194](#))
5. 《7 系列 FPGA Integrated Block for PCI Express LogiCORE IP 产品指南》([PG054](#))

6. 《Virtex-7 FPGA Integrated Block for PCI Express LogiCORE IP 产品指南》(PG023)
7. 《UltraScale 器件 Gen3 Integrated Block for PCI Express LogiCORE IP 产品指南》(PG156)
8. 《UltraScale+ Integrated Block for PCI Express LogiCORE IP 产品指南》(PG213)
9. 《Versal ACAP DMA and Bridge Subsystem for PCI Express 产品指南》(PG344)
10. 《Vivado Design Suite 用户指南：采用 IP integrator 设计 IP 子系统》(UG994)
11. 《Vivado Design Suite 用户指南：采用 IP 进行设计》(UG896)
12. 《Vivado Design Suite 用户指南：入门指南》(UG910)
13. 《Vivado Design Suite 用户指南：使用约束》(UG903)
14. 《Vivado Design Suite 用户指南：逻辑仿真》(UG900)
15. 《Vivado Design Suite 用户指南：Dynamic Function eXchange》(UG909)
16. 《ISE 到 Vivado Design Suite 移植指南》(UG911)
17. 《Vivado Design Suite 用户指南：编程和调试》(UG908)
18. 《Vivado Design Suite 用户指南：实现》(UG904)
19. 《AXI Interconnect LogiCORE IP 产品指南》(PG059)

## 修订历史

下表列出了本文档的修订历史。

章节	修订综述
<b>2021 年 4 月 29 日 4.1 版</b>	
常规更新	新增指向《Versal ACAP DMA and Bridge Subsystem for PCI Express 产品指南》(PG344) 的引用和链接以提供更多信息。
UltraScale+ 器件	更新受支持的器件。
<b>2020 年 9 月 21 日 4.1 版</b>	
常规更新	全文澄清。
串联配置	更新对 Dynamic Function eXchange 的部分重配置参考内容。
“Debug Options” 选项卡	新增调试选项。
定制 PIPE 仿真的参数	新增有关必需的 PIPE 仿真参数的指导信息。
附录 A: 应用软件开发	更新链接以提供附加的驱动程序信息。
<b>2019 年 11 月 22 日 4.1 版</b>	
串联配置	更新受支持的器件。
MSI-X 矢量表和 PBA (0x8)	新增 MSI-X 表偏移值和 PBA 表偏移值。
<b>2019 年 6 月 20 日 4.1 版</b>	
串联配置	更新受支持的器件。
DMA C2H 数据流传输	澄清 C2H 数据流传输描述符长度必需为 64 个字节的倍数。
IRQ 块寄存器 (0x2)	澄清 AXI Bridge 模式的 MSI-X 中断寄存器描述。
自定义和生成子系统	更新截屏。

章节	修订综述
“Basic” 选项卡	新增 “GT DRP Clock Selection” 选项和 “Data Protection” 选项。
“PCIe MISC” 选项卡	新增 “MSI RX PIN EN” 选项。
“PCIe DMA” 选项卡	移除 “Parity Checking” 选项（移至 “Basic” 选项卡下）。
附录 A: 应用软件开发	附录已从 “器件驱动程序” 重命名为 “应用软件开发”。
<b>2018 年 12 月 5 日 4.1 版</b>	
第 3 章: 产品规格	在 “器件最低要求” 表中: <ul style="list-style-type: none"> <li>· 新增受架构器件 Gen3 x16 (PCIe4) 支持的 -2 速度等级。</li> <li>· 为具有高带宽存储器 (HBM) 的 Virtex UltraScale+™ 器件新增 Gen4 链路速度详细信息。</li> <li>· 在 “H2C 通道” 部分中新增有关多个通道的行为的信息。</li> <li>· 新增有关 AXI4-Lite 从接口的访问限制的信息。</li> <li>· 更新 cfg_ext_read_received 描述。</li> </ul>
第 4 章: 利用子系统进行设计	在 “受支持的串联 Tandem PROM/PCIe 配置 (UltraScale+ 器件)” 表中更新量产支持详细信息。
第 7 章: 测试激励文件	<ul style="list-style-type: none"> <li>· 更新 “描述符旁路模式” 描述, 以反映 H2C 和 C2H 描述符具有 128 个字节的数据。</li> <li>· 新增 Tcl 命令, 用于对设计示例进行综合后仿真。</li> </ul>
第 6 章: 设计示例	新增用户 IRQ 设计示例
<b>2018 年 4 月 4 日 4.1 版</b>	
常规更新	澄清在 UltraScale+ 器件中对于 Bridge 模式暂不支持串联配置。
第 2 章: 概述	新增限制: 对于 7 系列, 从主机系统执行的 PCIe 访问必须限制为 1DW (4 个字节) 传输事务。
第 3 章: 产品规格	<p>新增有关 IRQ 模块配置组件 (传统中断、MSI 中断和 MSI-X 中断部分) 的澄清文本。</p> <p>在 “H2C 通道 0-3 AXI4-Stream 接口信号” 表以及 “C2H 通道 0-3 AXI4-Stream 接口信号” 表中进行编辑更新。</p> <p>在 “顶层接口信号” 表中新增 dma_bridge_resetsn 信号。</p> <p>更新寄存器名称: IRQ Block Channel Interrupt Pending (0x4C)</p> <p>新增有关含 HBM 的 Virtex UltraScale+ 器件 (PCIe4C) 的器件最低要求信息。</p>
第 4 章: 利用子系统进行设计	在 “受支持的串联 PROM/PCIe 配置 (UltraScale+ 器件)” 表中新增 Virtex UltraScale+ 器件。 为 7 系列 Gen2 系列器件新增共享逻辑支持。
器件驱动程序附录	在 “MSI 中断、MSI-X 中断和用户中断” 部分中新增澄清文本。
<b>2017 年 12 月 20 日 4.0 版</b>	
常规更新	<p>更新 “器件最低要求” 表中有关 Gen 3 x8 支持的信息。</p> <p>新增有关 h2c_dsc_byp_ctl[15:0] 和 c2h_dsc_byp_ctl[15:0] 端口描述的详细信息。</p> <p>为描述符旁路模式新增时序图。</p> <p>在 “PCIe 到 DMA 地址字段描述” 表中更新有关 11:8 位索引 (“Channel ID[3:0]” 字段) 的描述。</p> <p>在附录 “升级” 中新增 c_s_axi_supports_narrow_burst 参数。</p>

章节	修订综述
<b>2017 年 10 月 4 日 4.0 版</b>	
常规更新	<p>在本指南中已移除 PCIe AXI Bridge 模式操作，此操作已移至《AXI Bridge for PCI Express Gen3 Subsystem 产品指南》(PG194)。本文档 (PG195) 仅涵盖有关 DMA 模式操作的内容。</p> <p>在“串联配置”部分中，为 UltraScale+ 器件新增指示信息和器件支持信息，并为 UltraScale 器件新增器件支持信息。</p> <p>根据对应该版本的核的端口和参数更改，更新“升级”附录。</p> <p>新增附录 D “使用赛灵思虚拟线缆进行调试”。</p>
<b>2017 年 6 月 7 日 3.1 版</b>	
常规更新	<p>更新 [NUM_USR_INT-1:0] 位描述详细信息。</p> <p>更新 PCI Extended Tag 参数描述。</p> <p>在“产品规格”章节中新增有关 DMA C2H 和 H2C 传输的快速入门信息。</p>
<b>2017 年 4 月 5 日 3.1 版</b>	
常规更新	<p>更新驱动程序支持，Windows 驱动程序处于预量产阶段。</p> <p>更新标识符版本。</p> <p>新增 GUI 参数：Reset Source 和 MSI-X Implementation Location 和 AXI 未完成的传输事务。</p> <p>新增基于 Vivado IP integrator 的设计示例。</p> <p>更新“测试激励文件”章节中的“仿真”部分和“描述符旁路模式”部分。</p> <p>在“升级”附录中新增参数和端口。</p>
<b>2017 年 2 月 21 日 3.0 版</b>	
常规更新	<p>在“器件最低要求”表中更新 UltraScale+ 器件支持的速度等级。</p>
<b>2016 年 11 月 30 日 3.0 版</b>	
常规更新	<p>更新核名称，以反映 2 种核功能模式：AXI Bridge Subsystem for PCIe（仅限 UltraScale+）和 DMA Subsystem for PCIe（所有其它受支持的器件）。</p> <p>根据这 2 种功能模式可用的选项对“自定义和生成子系统”部分（第 4 章节）进行了组织。</p> <p>在 Vivado IDE 中新增“Debug Options”选项卡，以在核中启用调试选项。</p> <p>更新标识符版本。</p>
<b>2016 年 10 月 12 日 3.0 版</b>	
常规更新	<p>新增 Artix®-7 和 Zynq-7000 SoC 器件限制，仅限 7A15T 和 7A25T 不受支持。</p>

章节	修订综述
<b>2016 年 10 月 5 日 3.0 版</b>	
常规更新	<p>新增额外的器件系列支持。</p> <p>新增支持，以配合赛灵思 Gen2 Integrated Block for PCIe 核使用。</p> <p>在网上答复记录中新增性能数据。</p> <p>在“DMA 操作”部分中的“地址对齐”表和“长度粒度”表中更新数据路径宽度和限制。</p> <p>更新端口描述：</p> <ul style="list-style-type: none"> <li>· 新增针对奇偶校验端口的支持。</li> <li>· 新增针对“配置扩展”端口的支持。</li> </ul> <p>更新“寄存器空间”描述：</p> <ul style="list-style-type: none"> <li>· 更新标识符版本。</li> <li>· 新增 H2C SGDMA Descriptor Credits (0x8C)、C2H SGDMA Descriptor Credits (0x8C)、SGDMA Descriptor Credit Mode Enable (0x20)、SG Descriptor Mode Enable Register (0x24) 和 SG Descriptor Mode Enable Register (0x28)。</li> </ul> <p>更新 Vivado IP 目录描述 (2016.3)：</p> <ul style="list-style-type: none"> <li>· 更新“PCIe: BARs”选项卡、“PCIe: Misc”选项卡和“PCIe: DMA”选项卡。</li> <li>· 新增“Shared Logic”选项卡。</li> </ul> <p>新增“基本 Vivado 仿真”部分。</p> <p>新增“含描述符旁路模式的 AXI-MM 示例”部分。</p> <p>在“调试”附录中新增额外受支持的 7 系列评估板。</p>
<b>2016 年 6 月 8 日 2.0 版</b>	
常规更新	<p>标识符版本更新</p> <p>已添加 AXI4-Stream Writeback Disable Control 位文档记录</p>
<b>2016 年 4 月 6 日 2.0 版</b>	
初始赛灵思版本。	不适用

## 请阅读：重要法律提示

本文向贵司/您所提供的信息（下称“资料”）仅在对赛灵思产品进行选择和使用参考。在适用法律允许的最大范围内：(1) 资料均按“现状”提供，且不保证不存在任何瑕疵，赛灵思在此声明对资料及其状况不作任何保证或担保，无论是明示、暗示还是法定的保证，包括但不限于对适销性、非侵权性或任何特定用途的适用性的保证；且 (2) 赛灵思对任何因资料发生的或与资料有关的（含对资料的使用）任何损失或赔偿（包括任何直接、间接、特殊、附带或连带损失或赔偿，如数据、利润、商誉的损失或任何因第三方行为造成的任何类型的损失或赔偿），均不承担责任，不论该等损失或者赔偿是何种类或性质，也不论是基于合同、侵权、过失或是其它责任认定原理，即便该损失或赔偿可以合理预见或赛灵思事前被告知有发生该损失或赔偿的可能。赛灵思无义务纠正资料中包含的任何错误，也无义务对资料或产品说

明书发生的更新进行通知。未经赛灵思公司的事先书面许可，贵司/您不得复制、修改、分发或公开展示本资料。部分产品受赛灵思有限保证条款的约束，请参阅赛灵思销售条款：<https://china.xilinx.com/legal.htm#tos>；IP 核可能受赛灵思向贵司/您签发的许可证中所包含的保证与支持条款的约束。赛灵思产品并非为故障安全保护目的而设计，也不具备此故障安全保护功能，不能用于任何需要专门故障安全保护性能的用途。如果把赛灵思产品应用于此类特殊用途，贵司/您将自行承担风险和责任。请参阅赛灵思销售条款：<https://china.xilinx.com/legal.htm#tos>。

### 关于与汽车相关用途的免责声明

如将汽车产品（部件编号中含“XA”字样）用于部署安全气囊或用于影响车辆控制的应用（“安全应用”），除非有符合 ISO 26262 汽车安全标准的安全概念或冗余特性（“安全设计”），否则不在质保范围内。客户应在使用或分销任何包含产品的系统之前为了安全的目的全面地测试此类系统。在未采用安全设计的条件下将产品用于安全应用的所有风险，由客户自行承担，并且仅在适用的法律法规对产品责任另有规定的情况下，适用该等法律法规的规定。

### 版权声明

© Copyright 2016-2021 赛灵思公司版权所有。“赛灵思”、“赛灵思”徽标、Alveo、“Artix”、“Kintex”、“Spartan”、“Versal”、“Virtex”、“Vivado”、“Zynq”以及本文提到的其它指定品牌均为赛灵思在美国及其它国家或地区的商标。“PCI”、“PCIe”和“PCI Express”均为 PCI-SIG 拥有的商标，且经授权使用。“AMBA”、“AMBA Designer”、“Arm”、“ARM1176JZ-SV”、“CoreSight”、“Cortex”、“PrimeCell”、“Mali”和“MPCore”为 Arm Limited 在欧盟及其它国家或地区的注册商标。所有其它商标均为各自所有方所属财产。