

Arria 10 Avalon-MM DMA 接口 PCIe 解决方案用户指南

针对 Intel® Quartus® Prime 设计套件的更新: **16.1**



UG-01145_avmm_dma | 2016.10.31

官网最新文档: [PDF](#) | [HTML](#)



内容

1. 数据表	5
1.1. Arria®10 Avalon-MM DMA 接口 PCIe 数据表.....	5
1.2. 特性.....	6
1.3. 比较 Avalon-ST, Avalon-MM 和支持 DMA 的 Avalon-MM 接口.....	7
1.4. 发布信息	9
1.5. 器件系列支持	9
1.6. 设计实例.....	9
1.7. 调试功能.....	10
1.8. IP Core 验证	10
1.8.1. 兼容性测试环境	10
1.9. 性能和资源利用	10
1.10. 建议的速度等级	10
1.11. 创建 PCI Express 设计.....	11
2. Avalon-MM DMA 入门	13
2.1. 生成 Testbench	14
2.1.1. 了解仿真已生成文件	15
2.1.2. 了解仿真日志文件生成.....	16
2.2. 在 ModelSim 中仿真实例设计.....	16
2.3. 运行门级 (Gate-Level) 仿真.....	16
2.4. 生成综合文件.....	16
2.5. 创建 Quartus Prime 项目.....	17
2.6. 编译设计.....	18
2.7. 单独例化时描述符控制器的连接性.....	18
3. 参数设置.....	19
3.1. 参数	19
3.1.1. 基于接口类型的 RX 缓冲分配可用选择.....	20
3.2. Arria 10 Avalon-MM 设置.....	20
3.3. 基地址寄存器 (BAR) 设置	21
3.4. 器件识别寄存器	22
3.5. PCI Express 和 PCI 性能参数.....	22
3.5.1. 器件性能	22
3.5.2. 错误报告.....	23
3.5.3. 链路性能	24
3.5.4. MSI 和 MSI-X Capabilities	24
3.5.5. 插槽性能	25
3.5.6. 电源管理.....	25
3.5.7. 供应商指定扩展性能 (VSEC)	26
3.6. 配置, 调试, 和扩展项.....	26
3.7. PHY 特性	27
3.8. Arria10 设计实例.....	27
4. Arria 10 器件中 Hard IP 的物理布局.....	28
4.1. Gen1, Gen2, Gen3 数据速率的 Channel 和管脚布局.....	30

4.2. Gen1 和 Gen2 数据速率的 Channel 布局和 fPLL 使用	32
4.3. Gen3 数据速率的 Channel 布局, 以及 fPLL 和 ATX PLL 的使用.....	34
4.4. PCI Express Gen3 Bank 组使用限制.....	36
5. IP 内核接口	37
5.1. Arria10 DMA Avalon-MM DMA 接口到应用层.....	37
5.1.1. 读 DMA Avalon-MM 主端口	40
5.1.2. 写 DMA Avalon-MM 主端口	40
5.1.3. RX Master 模块	41
5.1.4. TX Slave 模块	42
5.1.5. 32-Bit 非突发 Avalon-MM 控制寄存器访问 (CRA) Slave 信号	43
5.1.6. Avalon-ST 描述符控制接口单独例化时	44
5.1.7. 描述符控制器接口内部例化时.....	46
5.2. 时钟信号.....	47
5.3. 复位, 状态, 和链路训练信号.....	47
5.4. 端点的 MSI 中断	50
5.5. 硬核 IP 重配置接口	51
5.6. 物理层接口信号	52
5.6.1. 串行数据信号.....	52
5.6.2. PIPE 接口信号	53
5.7. 测试信号.....	55
5.8. Arria 10 开发套件导管接口.....	56
6. 寄存器.....	58
6.1. 配置空间寄存器与 PCIe 规范的一致性.....	58
6.2. Type 0 配置空间寄存器	61
6.3. Type 1 配置空间寄存器.....	62
6.4. PCI Express 性能结构.....	62
6.5. Intel 定义的 VSEC 寄存器.....	65
6.6. CvP 寄存器.....	66
6.7. 不可纠正的内部错误掩码寄存器.....	68
6.8. 不可纠正的内部错误状态寄存器.....	68
6.9. 可纠正的内部错误掩码寄存器.....	69
6.10. 可纠正的内部错误状态寄存器.....	69
6.11. DMA 描述符控制器寄存器.....	70
6.11.1. 读 DMA 描述符控制器寄存器.....	71
6.11.2. 写 DMA 描述符控制器寄存器.....	72
6.11.3. 读 DMA 和写 DMA 描述符列表格式.....	73
6.11.4. 读 DMA 实例	74
6.11.5. 软件编程以同时读和写 DMA	78
6.12. 控制寄存器访问 (CRA) Avalon-MM 从端口	79
7. Arria 10 复位和时钟.....	83
7.1. Hard IP for PCI Express IP 内核及应用层复位序列.....	84
7.2. 时钟.....	85
7.2.1. 时钟域	85
7.2.2. 时钟摘要.....	87



8. 错误处理	88
8.1. 物理层错误	88
8.2. 数据链路层错误	88
8.3. 事务层错误	89
8.4. 错误报告和数据中毒	90
8.5. 不可纠正和可纠正的错误状态位	90
9. IP Core 体系结构	92
9.1. 顶层接口	93
9.1.1. Avalon-MM DMA 接口	93
9.1.2. 时钟和复位	93
9.1.3. 中断	93
9.1.4. PIPE	94
9.2. 数据链路层	94
9.3. 物理层	96
9.4. Arria10 Avalon-MM DMA 用于 PCI Express	98
9.4.1. 了解内部 DMA 描述符控制器	98
9.4.2. 了解外部 DMA 描述符控制器	100
10. 设计实现	102
10.1. 进行管脚约束以分配 I/O 标准到串行数据管脚	102
10.2. 建议的复位序列以避免链路训练问题	102
10.3. 创建 SignalTap II 调试文件以匹配设计层次	103
10.4. SDC 时序约束	104
A. 传输层数据包 (TLP) 头格式	105
A.1. 带有数据负载的 TLP 数据包	107
B. Arria 10 Avalon-MM DMA 接口 PCIe 解决方案用户指南存档	109
C. 修订历史	110
C.1. 具有 DMA 的 Avalon-MM 接口文档修订历史	110

1. 数据表

1.1. Arria®10 Avalon-MM DMA 接口 PCIe 数据表

Intel® Arria 10 FPGA 包括一个用于 PCI Express® 的可配置，加强协议栈，并与 *PCI Express Base Specification 3.0* 兼容。

The Arria®10 Hard IP for PCI Express 及 Avalon® Memory-Mapped (Avalon-MM) DMA 接口删除了 PCIe 协议相关的复杂性。例如，IP 内核处理 TLP 编码和解码。此外，IP 内核包括 Read DMA 和 Write DMA 引擎。如果您已构架了自己的 DMA 系统与 Avalon-MM 接口，或许希望继续使用。然而，您也可受惠于已实现 DMA 引擎的简便。该协议的新用户应该使用此 IP 内核。该 variant 在 Qsys 中用于 128-和 256-bit 接口连接到应用层。Avalon-MM 接口和 DMA 引擎实现于 FPGA 软逻辑。

图 1. 支持 Avalon-MM DMA 接口的 Arria10 PCIe Variant

下图显示了此 variant 的高级模块和接口连接。

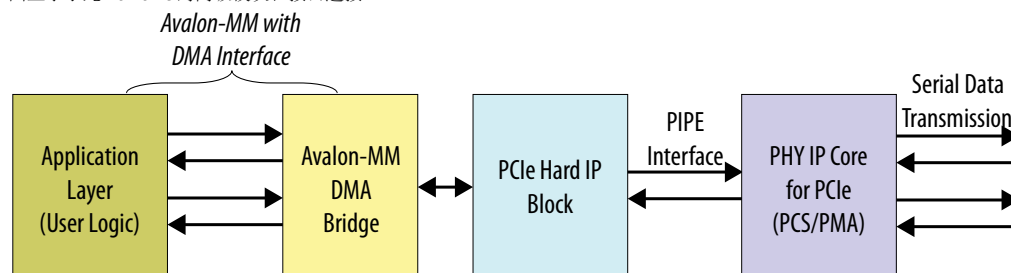


表 1. PCI Express 数据吞吐量

下表罗列了 PCI Express 链路 Gen1, Gen2, 和 Gen3 的 2, 4, 以及 8 lane 的集成带宽。根据协议规定，Gen1 是 2.5 千兆传输每秒，Gen2 是 5.0 千兆传输每秒，和 Gen3 的 8.0 千兆传输每秒。本表格提供了单个发送 (TX) 或接收 (RX) channel 的带宽。这些数字加倍双工操作。Gen1 和 Gen2 使用 8B/10B 编码，并产生 20% 的开销，相反，Gen3 使用 128b/130b 编码，仅产生 1.5% 的开销。

以千兆每秒 (Gbps) 为单位。

	链路宽度		
	×2	×4	×8
PCI Express Gen1 (2.5 Gbps)	N/A	N/A	16 Gbps
PCI Express Gen2 (5.0 Gbps)	N/A	16 Gbps	32 Gbps
PCI Express Gen3 (8.0 Gbps)	15.75 Gbps	31.51 Gbps	63Gbps

相关链接

- [Arria 10 Avalon-MM DMA 接口 PCIe 解决方案用户指南存档 \(第 109 页\)](#)
- [Intel FPGA IP 内核的简介](#)
提供有关所有 Intel FPGA IP 内核的一般信息，包括参数化、生成、更新和仿真 IP 内核。
- [Creating Version-Independent IP and Qsys Simulation Scripts](#)
创建不需要对软件进行手动更新和不需要 IP 版本升级的仿真脚本。
- [Project Management Best Practices](#)
提供关于您的工程和 IP 文件的高效管理和可移植性指南。
- [AN 690: PCI Express Avalon-MM DMA Reference Design](#)
关于说明使用内部存储器的 chaining DMA 性能的参考设计。
- [PCI Express Base Specification 3.0](#)

1.2. 特性

Quartus® Prime 16.1 发布中的新特性:

- 把 DMA 最大传输量从 128-和 256-bit 提高到 1 megabyte (MB)。
- 为 Phase2 和 Phase3 远端 TX 均衡选择所请求预设添加了参数。
- 时序模块最终用于大多数 Arria 10 器件封装。但某些具有扩展温度范围的军事和汽车速度等级除外。

Arria10 Avalon-MM DMA for PCI Express 支持下列特性:

- 完成协议栈包括交易，数据链路，和作为硬核 IP 的物理层实现。
- Gen1 x8, Gen2 x4, Gen2 x8, Gen3 x2, Gen3 x4, Gen3 x8 端点的本地支持。当插入较少的链路宽度或更改为另一个较大链路速率时此 variant 向下运行。
- Dedicated 16 KB receive buffer.
- 对具有嵌入式 DMA 的 128-或 256-bit Avalon-MM 接口到应用层高达 Gen3 x8 数据速率支持。
- 支持 Avalon-MM 接口到应用层的 32-或 64-bit 寻址。
- Qsys 设计实例演示参数化，设计模块，及连接性。
- 扩展的信用分配设置更好的优化基于应用类型的 RX 缓冲空间。
- 可选的 end-to-end 循环冗余代码 (ECRC) 生成和查看，以及高可靠性应用的错误报告 (AER)。
- 简单易用:
 - 灵活的配置。
 - 充足的片上资源储存和有保障的时序收敛。
 - 无需许可证。

表 2. 比较连接到应用层的 128-和 256-Bit 带有 DMA 的 Avalon-MM 接口

特性	128-Bit 接口	256-Bit 接口
Gen1	x8	不支持
Gen2	x4, x8	x8
继续...		



特性	128-Bit 接口	256-Bit 接口
Gen3	x4	x4, x8
根端口	不支持	支持
支持的 Tag 数目	16	16 或 256
最大描述符	1 MB	1 MB
最大有效负载量	128 或 256	128 或 256
立即写 ⁽¹⁾	不支持	支持

1.3. 比较 Avalon-ST, Avalon-MM 和支持 DMA 的 Avalon-MM 接口

表 3. 全部 Hard IP for PCI Express IP Core 的特性比较

此表格比较三种主流的 Hard IP for PCI Express IP Core。请参阅 *Arria 10 支持 SR-IOV 的 Avalon-ST 接口 PCIe 解决方案用户指南* 了解此 variant 的特性。

特性	Avalon-ST 接口	Avalon-MM 接口	Avalon-MM DMA
IP Core 许可证	免费	免费	免费
本地端点	支持	支持	支持
根端口	支持	支持	支持
Gen1	x1、x2、x4、x8	x1、x2、x4、x8	x8
Gen2	x1、x2、x4、x8	x1、x2、x4、x8	x4, x8
Gen3	x1、x2、x4、x8	x1、x2、x4	x2, x4, x8
64-bit 应用层接口	支持	支持	不支持
128-bit 应用层接口	支持	支持	支持
256-bit 应用层接口	支持	支持	支持
最大有效负载量	128、256、512、1024、2048 bytes	128, 256 bytes	128, 256 bytes
支持 non-posted 请求的 tag 数	256	8	16 或 256
自动处理无序完成 (对应用层透明)	不支持	支持	支持
自动处理跨 4 KB 地址边界请求 (对应用层透明)	不支持	支持	支持
PIPE 接口信号的极性反转	支持	支持	支持
MSI 请求数	1、2、4、8、16、或 32	1、2、4、8、16、或 32	1、2、4、8、16、或 32
MSI-X	支持	支持	支持
Legacy 中断	支持	支持	支持
扩展 ROM	支持	不支持	不支持
PCIe 分岔 (PCIe bifurcation)	不支持	不支持	不支持

⁽¹⁾ Immediate Write 为发送 Write TLP 上游提供快速机制。描述符储存 32-bit 有效负载并替代描述符 Source Low Address 域。

表 4. 比较全部 Hard IP for PCI Express IP Core 的 TLP 支持

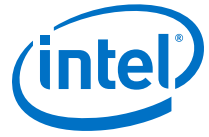
此表格比较了 Hard IP for PCI Express IP 内核 variant 可发送的 TLP 类型。每个条目表示该 TLP 类型是否可通过端点 (EP)，根端口 (RP)，或 (EP/RP) 两者皆可 (发送)。对于 Avalon-MM DMA 接口，软件应用编程一个描述符控制器以指定主机和 IP 存储器之间的 DMA 传输。Read DMA Avalon-MM 主端口和 Write DMA Avalon-MM 主端口分别发送读和写 TLP。可选的 TX Slave 模块支持信号，non-bursting Memory Write TLP，发送状态更新到主机。

TLP (发送支持)	Avalon-ST 接口	Avalon-MM 接口	Avalon-MM DMA
存储器读请求 (Mrd)	EP/RP	EP/RP	EP/RP (Read DMA Avalon-MM Master)
存储器读锁定请求 (MRdLk)	EP/RP		不支持
存储器写请求 (MWr)	EP/RP	EP/RP	EP/RP (Write DMA Avalon-MM Master) (TX Slave - 可选)
I/O 读请求 (IOrd)	EP/RP	EP/RP	不支持
I/O 写请求 (IOWr)	EP/RP	EP/RP	不支持
Config Type 0 读请求 (CfgRd0)	RP	RP	不支持
Config Type 0 写请求 (CfgWr0)	RP	RP	不支持
Config Type 1 读请求 (CfgRd1)	RP	RP	不支持
Config Type 1 写请求 (CfgWr1)	RP	RP	不支持
消息请求 (Msg)	EP/RP	不支持	不支持
带数据的消息请求 (MsgD)	EP/RP	不支持	不支持
完成 (Cpl)	EP/RP	EP/RP	EP/RP (Read & Write DMA Avalon-MM Masters)
带数据的完成 (CplD)	EP/RP	不支持	EP/RP (Read & Write DMA Avalon-MM Masters)
完成已锁定 (CplLk)	EP/RP	不支持	不支持
带数据的完成锁定 (CplDLk)	EP/RP	不支持	不支持
提取以及添加 AtomicOp 请求 (FetchAdd)	EP	不支持	不支持

Arria10 Avalon-MM DMA Interface for PCIe 解决方案用户指南解释如何使用此 IP 内核而未介绍 PCI Express 协议。但这两方面必然存在重叠之处，所以使用本文档时必须结合对 PCI Express Base Specification 的了解。

相关链接

- [Arria 10 Avalon-MM Interface for PCIe Solutions User Guide](#)
针对不具有 DMA 的 Avalon-MM 接口。
- [Arria 10 Avalon-ST Interface for PCIe Solutions User Guide](#)
针对 Avalon-ST 接口。



- [Arria 10 Avalon-ST Interface with SR-IOV PCIe Solutions User Guide](#)
关于具有单个根 I/O 的 Avalon-ST 接口虚拟化 (SR-IOV)。
- [PCI Express Base Specification 3.0](#)

1.4. 发布信息

表 5. **Hard IP for PCI Express** 发布信息

项目	说明
版本	16.1
发布日期	2016 年 10 月
订购代码	无需订购代码
产品 ID	由于此 IP 内核不需要许可证，所以也不需要产品 ID 和供应商 ID。
供应商 ID	

Intel 通过当前版本 Quartus Prime 软件编译之前版本中的每个 IP 内核，验证是否此 IP 内核曾用于之前的发布。Intel 通过 *Intel IP Release Notes* 报告此次验证中的任何异常或者通过 Quartus Prime IP 更新工具阐明出现的异常。Intel 不验证编译比前一发布更早版本中的 IP 内核。

相关链接

- [Intel FPGA IP Release Notes](#)
提供关于 Intel FPGA IP 内核目前及之前版本的发布笔记。
- [知识库中查看 Errata for the Arria 10 Hard IP for PCI Express IP Core](#)

1.5. 器件系列支持

表 6. **器件系列支持**

器件系列	支持
Arria 10	最后。使用最终时序模型验证 IP 内核。此 IP 内核符合该器件系列的所有功能性和时序性要求，并可用于生产设计。
其他器件系列	请参阅 <i>Intel's PCI Express IP Solutions</i> 网页了解关于其他器件系列的支持信息。

相关链接

[PCI Express Web Page](#)

1.6. 设计实例

Qsys 设计实例可用于 Arria10 Avalon-MM DMA for PCI Express IP Core。可以从 `<install_dir>/ip/altera/altera_pcie/altera_pcie_a10_ed/example_design/a10` 目录下载。

相关链接

[Avalon-MM DMA 入门 \(第 13 页\)](#)



1.7. 调试功能

调试功能可以观察和控制 IP，从而更快调试系统级问题。

1.8. IP Core 验证

为确保符合 PCI Express 规范，Intel 进行广泛验证。此仿真环境使用的多种 testbench 中包含驱动 PCI Express 链路接口的工业标准总线功能模型（BFM）。Intel 在仿真环境中执行以下测试：

- 直接的，伪随机激励测试应用层接口，配置空间，以及所有类型和大小的 TLP
- Error injection 测试链路，TLP，和数据链路层数据包（DLLP）中的注入错误，并查找正确的响应
- 针对测试清单中各项的 PCI-SIG® Compliance Checklist 测试
- 大范围测试流量码型的随机测试

经请求后，可采用 Intel 提供的设计实例测试 PCB 及 PCI-SIG 的完成合规基线板测试（CBB 测试）。

1.8.1. 兼容性测试环境

Intel 已进行了重要的硬件测试以确保解决方案可靠。此外，Intel 使用来自多个制造商的母板和 PCI Express 交换开关对每次发布进行内测。每个 IP 内核发布都进行了全面的 PCI-SIG 兼容测试。

1.9. 性能和资源利用

由于 PCIe 协议栈实现于加强逻辑，所以使用无内核器件资源（无 ALM 和无嵌入式存储器）。

支持 DMA 的 Avalon-MM Arria10 variant 在软逻辑中有一个已实现的 Avalon-MM DMA 桥，并作为前端运行于加强协议栈。以下表格显示了使用当前版本 Quartus Prime 软件针对 Arria10 器件已选配置的常见期望器件资源利用情况。除 M20K 存储器块之外，ALM 和逻辑寄存器个数极近 50。

表 7. Arria10 Avalon-MM DMA for PCI Express 的性能和资源利用

数据速率，Lane 数，及接口宽度	ALMs	M20K 存储器块	逻辑寄存器
Gen2 x8 128	12700	19	22300
Gen3 x8 256	18000	47	31450

相关链接

[运行 Fitter](#)

关于 Fitter 约束的信息。

1.10. 建议的速度等级

建议的速度等级是成品 Arria 10 器件的待定表征。

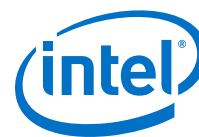


表 8. Arria 10 关于所有支持 DMA 的 Avalon-MM 宽度和频率的建议

Lane 速率	链路宽度	接口宽度	应用时钟频率 (MHz)	推荐的速度等级
Gen1	×8	128-bit	125	- 1、- 2、- 3
Gen2	×4	128-bit	125	- 1、- 2、- 3
	×8	128-bit	250	- 1, - 2
Gen3	×2	128-bit	125	- 1、- 2、- 3
	×4	128-bit	250	- 1, - 2
	×8	256-bit	250	- 1, - 2

相关链接

- [Intel FPGA Software Installation and Licensing](#)
提供关于 Intel FPGA 软件安装和许可的详细信息。
- [运行综合](#)
影响时序收敛的设置。

1.11. 创建 PCI Express 设计

选择最符合您设计要去的 PCIe variant。

- 您的设计是端点还是根端口？
 - 打算实现的生成是什么？
 - 打算实现的链路宽度为多少？
 - 应用程序要求多少带宽？
 - 设计中是否需要 Configuration via Protocol (CvP)？
1. 为此 variant 选择参数。
 2. 对于 Arria10 器件，可使用 GUI 组件中新的 **Example Design** 选项卡生成指定的设计。然后，仿真此实例并下载到 Arria 10 FPGA 开发套件。请参阅 *Arria 10 PCI Express IP Core Quick Start Guide* 了解详情。
 3. 所有器件中，都可使用 Intel 提供的设计实例进行仿真，全部 Intel 静态 PCI Express 设计实例都可在<install_dir>/ip/altera/altera_pcie/altera_pcie_<dev>_ed/example_design/<dev>中找到。或者，生成一个与您参数设置匹配的设计实例，还是创建一个仿真模型，并使用自定义或第三方 BFM。Qsys Generate menu 生成仿真模型 generates simulation models. Intel 支持 ModelSim® - Intel FPGA Edition 用于所有 IP。PCIe 内核支持 Aldec RivieraPro, Cadence NCsim, Mentor Graphics ModelSim, 和 Synopsys® VCS 及 VCS-MX 仿真器。

Intel testbench 和根端口或端点 BFM 为连接此 variation 的应用层逻辑基本测试提供简易方法，但，testbench 和根端口 BFM 并不会替代整个验证环境。要完整测试您的应用，Intel 建议使用商业版 PCI Express 验证 IP 和工具，或者进行您自己的扩展硬件测试，或者两者都使用。
 4. 使用 Quartus Prime 软件编译设计。如果您的设计版本与正在运行的 Quartus Prime 软件版本不匹配，请重新生成您的 PCIe 设计。



5. 把您的设计下载到 Intel 开发板或您的 PCB 中。点击以下 *All Development Kits* 链接查看 Intel 开发板列表。
6. 测试硬件。可使用 Intel SignalTap® 逻辑分析仪或第三方协议分析仪进行行为观测。
7. 用您的应用层逻辑替换 Intel testbench 中的应用层逻辑。然后重复步骤 3 – 6。Intel testbench 中，PCIe 内核通常称为 DUT（被测器件）。应用层逻辑通常称为 APPS。

相关链接

- [参数设置](#) (第 19 页)
- [Avalon-MM DMA 入门](#) (第 13 页)
- [全部开发套件](#)
- [Intel Wiki PCI Express](#)

要完成设计实例并有助于创建新项目或指定功能，如与 PCI Express 相关的 MSI 或 MSI-X。Intel 应用工程师定期更新内容并添加新的实例设计。这些实例有助于设计者从 Intel PCI Express IP 内核获得更多产出并缩短面市时间。Intel 维基 (Wiki) 网页中的设计实例为您开发自己的设计提供使用的指导。但是，Intel 不保证 Intel 维基中的内容。



2. Avalon-MM DMA 入门

可从<install_dir>/ ip/altera/altera_pcie/altera_pcie_a10_ed/example_design/a10 目录下载 Qsys 设计实例，
ep_g3x8_avmm256_integrated.qsys

注意: 本实例设计说明为生成仿真和综合文件提供指导，但并不生成下载到硬件时的所有必要文件。前面章节中介绍的 *Arria 10 PCI Express Quick Start Guide* 包含将设计下载到 Arria 10 GX FPGA 开发套件时的所有必要文件。

设计实例包含以下组件：

Avalon-MM DMA for PCI Express

此 IP 内核包含高效 DMA Read 和 DMA Write 模块。DMA Read 和 Write 模块使用突发数据传输器在 PCI Express 地址域和 Avalon-MM 地址域之间高效地移动大块数据模块。基于您的所选配置，DMA Read 和 DMA Write 模块使用 128-或 256-bit Avalon-MM 数据通路。

除高性能数据传输以外，DMA Read 和 DMA Write 模块确保 PCI 链路上的要求符合 *PCI Express Base Specification, 3.0*。DMA Read 和 DMA Write 引擎还运行下列功能：

- 将原请求分成多个请求以避免跨 4KByte 边界。
- 将原请求分成多个请求以确保最大负载等同于或小于写请求的最大负载及读请求的最大读请求大小。
- 当原请求划分成符合读和写请求大小时支持无序完成。

使用 DMA Read 和 DMA Write 模块，您能指定描述符条目表中输入大型负载。

片上存储器 IP 内核

此 IP 内核储存 DMA 数据。该存储器的数据宽度是 256-bit。

描述符控制器

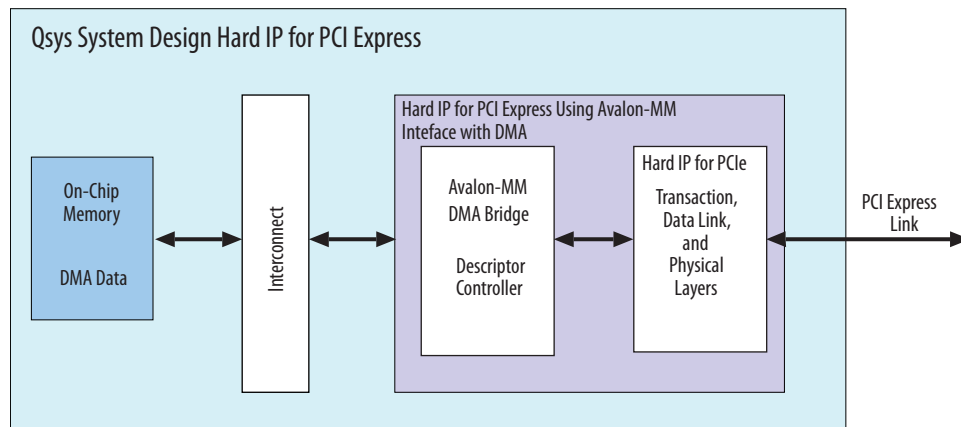
描述符控制器管理 Read DMA 和 Write DMA 模块。主机软件根据描述符表格所在的位置编程描述符控制器内部寄存器。描述符控制器指示读 DMA 模块将整个表格复制到其内部 FIFO。然后将表格条目推入 DMA 读或 DMA 写模块进行数据传输。描述符控制器还通过 Avalon-MM TX 从端口发送 DMA 状态下游。

本设计实例中，描述符控制器参数，**Instantiate internal descriptor controller** 是开启的。因而，如下图所示，描述符控制器被集成到 Avalon-MM 桥中。将描述符控制器嵌入 Avalon-MM DMA 桥中从而简化设计。若打算用个人实现代替描述符控制器 IP 内核，在参数化 IP 内核时就请勿打开参数编辑器中的 **Instantiate internal descriptor controller**。

描述符控制器具有下列特性：

- 单个双工 channel。
- 1 个 dword (4 bytes) 的最小传输量。
- 最大传输量为 1 MB - 4 bytes。
注意: 尽管描述符控制器支持的最大传输量为 (1 MB - 4 bytes)，但本设计实例中的片上存储器较小。因此，本设计实例无法处理最大传输量。
- 仅端点。
- DMA 完成传输时，通过生成一个中断向主机软件提供状态。

图 2. Arria 10 Avalon-MM DMA for PCI Express 框图



设计实例限制

本设计实例旨在显示 DMA 基本功能性，并不能替代用作强健检测 testbench。修改 Testbench 以创建更强健的激励可能导致错误的 testbench 行为。

相关链接

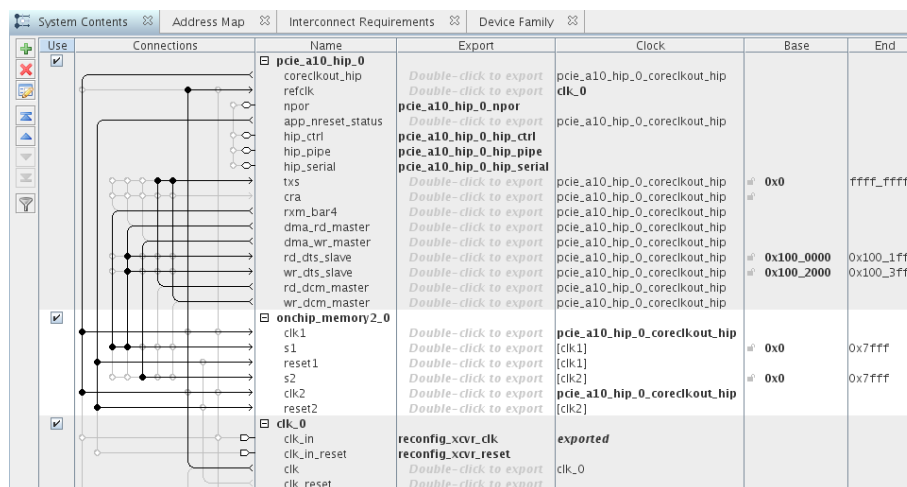
- [Arria10 Avalon-MM DMA 用于 PCI Express \(第 98 页\)](#)
- [DMA 描述符控制器寄存器 \(第 70 页\)](#)

2.1. 生成 Testbench

1. 从安装目录: `<install_dir>/ip/altera/altera_pcie/altera_pcie_a10_ed/example_design/a10/` 把设计实例, `ep_g3x8_avmm256_integrated.qsys` 复制到您的工作目录。
2. 键入下列命令, 开启 Qsys
`qsys-edit`
3. 打开 `ep_g3x8_avmm256_integrated.qsys`。



图 3. Arria 10 Avalon-MM DMA for PCI Express Qsys System Design



- 打开 `ep_g3x8_avmm256_integrated.qar` 并接受或修改 **Restore Archived Project** 对话框中指定的目录路径。点击 **OK**。
- 点击 **Generate > Generate Testbench System**。出现 **Generation** 对话框。
- 指定以下参数：

表 9. 在生成对话框中指定参数

参数	值
Testbench 系统	
Create testbench Qsys system	Standard, BFM's for standard Qsys interfaces
Create testbench simulation model	Verilog
Allow mixed-language simulation	该选项可以保持关闭。
输出目录	
Testbench	<working_dir>/ep_g3x8_avmm256_integrated_tb

- 点击 **Generate**。
Qsys 生成 testbench。

2.1.1. 了解仿真已生成文件

表 10. Qsys 生成输出文件

目录	说明
<working_dir>/ ep_g3x8_avmm256_integrated_tb/ ep_g3x8_avmm256_integrated_tb/	具有 testbench 全部组件的目录。还包含以下文件： <ul style="list-style-type: none"> 罗列所需仿真文件的 Simulation Package Descriptor File (.spd) 介绍 testbench 中各种文件的 Comma-Separated Value File (.csv)
<working_dir>/ ep_g3x8_avmm256_integrated_tb/ ep_g3x8_avmm256_integrated_tb/sim/ <cad_vendor>/	包含用于 Aldec, Cadence, Mentor 的 testbench 子目录，以及具有所要求的库和仿真脚本的 Synopsys 仿真工具。

2.1.2. 了解仿真日志文件生成

开启 Quartus II 14.0 软件发布，仿真自动在您的仿真目录创建一个日志文件，`altpcie_monitor_<dev>_dlhip_tlp_file_log.log`。

表 11. 采样仿真日志文件条目

时间	TLP 类型	负载(Bytes)	TLP 头 (Header)
17989 RX	CfgRd0	0004	04000001_0000000F_01080008
17989 RX	MRd	0000	00000000_00000000_01080000
18021 RX	CfgRd0	0004	04000001_0000010F_0108002C
18053 RX	CfgRd0	0004	04000001_0000030F_0108003C
18085 RX	MRd	0000	00000000_00000000_0108000C

2.2. 在 ModelSim 中仿真实例设计

- 在终端，将目录更改为`<workingdir>/pcie_g3x8_integrated_tb/ep_g3x8_avmm256_integrated_tb/sim/mentor`。
- 开启 ModelSim® 仿真器。
- 在终端窗口中输入下列命令，以运行仿真
 - `do msim_setup.tcl`
 - `ld_debug`
`ld_debug` 命令编译所有设计文件并细化顶层设计，无需优化。
 - `run -all`

此仿真运行以下操作：

- 初始化链路后各种配置访问
- 设置 DMA 控制器以读取来自事务层直接 BFM 共享存储器的数据
- 设置 DMA 控制器以把相同数据写入事务层直接 BFM 共享存储器
- 数据比较和失配报告

2.3. 运行门级 (Gate-Level) 仿真

PCI Express testbench 在寄存器传输级 (RTL) 运行仿真。但也可创建您自己的门级 (gate-level) 仿真。请联络您的 Intel 销售代表以获得如何从 RTL testbench 创建门级 (gate-level) 仿真的说明及实例演示。

2.4. 生成综合文件

- Generate** 菜单中，选择 **Generate HDL**。
- 要 **Create HDL design files for synthesis**，选择 **Verilog**。



所有其他项可保持默认设置。

3. 点击 **Generate** 生成用于综合的文件。
4. 完成生成时，点击 **Finish**。

相关链接

[What assignments do I need for a PCIe Gen1, Gen2 or Gen3 design that targets an Arria 10 ES2, ES3 or production device?](#)

2.5. 创建 Quartus Prime 项目

使用 New Project Wizard 创建一个新的 Quartus Prime 项目，可有助于为项目指定工作目录，分配项目名称，及指定顶层设计实体名称。

1. 点击 Quartus Prime 文件菜单上，选择 **New Project Wizard**，然后点选 **Next**。
2. 在 **New Project Wizard** 中点击 **Next: Introduction**（如果在此前已关闭了介绍，此时就不会出现。）
3. 在 **Directory, Name, Top-Level Entity** 页面，输入以下信息：
 - a. 关于 **What is the working directory for this project**，请浏览 `<project_dir>/ep_g3x8_avmm256_integrated/`。
 - b. 关于 **What is the name of this project?** 请浏览 `<project_dir>/ep_g3x8_avmm256_integrated/synth` 目录并选择 `ep_g3x8_avmm256_integrated.v`。
 - c. 点击 **Next**。
4. 关于 **Project Type** 选择 **Empty project**。
5. 点击 **Next**。
6. 在 **Add Files** 页面，将 `<project_dir>/ep_g3x8_avmm256_integrated/synth/ep_g3x8_avmm256_integrated.qip` 添加到 QuartusPrime 项目中。
7. 点击 **Next** 以显示 **Family & Device Settings** 页。
8. 在 **Device** 页面，选择下列目标器件系列及选项：
 - a. 在 **Family** 列表中选择 **Arria 10 (GX/SX/GT)**。
 - b. 在 **Devices** 列表中，选择 **All**。
 - c. 在 **Available devices** 列表中，选择正确的器件。为 Arria 10 GX FPGA 开发套件选择 **10AX115S2F45I1SG**。
9. 点击 **Next** 关闭该页，且显示 **EDA Tool Settings** 页。
10. 从 **Simulation** 列表中，选择 **ModelSim**。从 **Format** 列表中，选择您打算用于仿真的 HDL 语言。
11. 点击 **Next** 以显示 **Summary** 页面。
12. 查看 **Summary** 页确保全部信息输入正确。
13. 点击 **Finish**。
14. 保存您的项目。

2.6. 编译设计

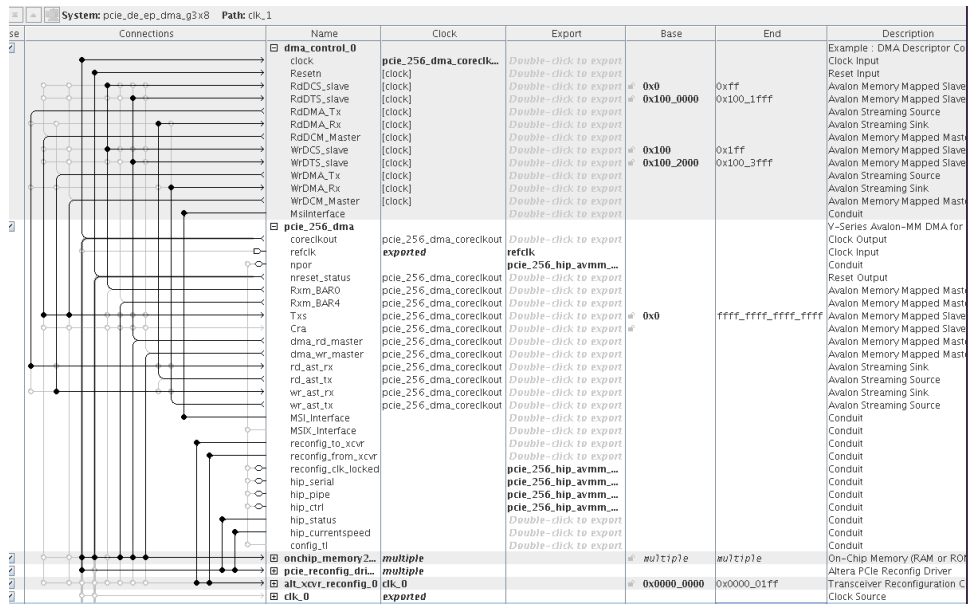
1. 在 Quartus Prime “Processing” 菜单上，点击 **Start Compilation**。
2. 编译后，展开编译报告中的 **TimeQuest Timing Analyzer** 文件夹。注意编译报告中时序约束是否都已实现。

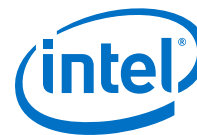
如果您的设计在一开始就不符合时序约束，则可使用 Design Space Explorer 找出适合您设计的优化 Fitter 设置。要使用 Design Space Explorer，请在工具菜单上点击 **Launch Design Space Explorer**。

2.7. 单独实例化时描述符控制器的连接性

Qsys 设计实例框图介绍如何把外部描述符控制器连接到具有 Avalon-MM DMA interface 接口的 Hard IP for PCI Express。此设计实例在<install_dir>/ip/altera/altera_pcie/altera_pcie_hip_256_avmm/example_design/<dev>中。

图 4. 外部描述符控制器的连接性





3. 参数设置

3.1. 参数

本章节为所有 Arria 10 Hard IP for PCI Express IP 参数提供参考。

表 12. 系统设置

参数	值	说明
Application Interface Type	Avalon-ST Avalon-MM Avalon-MM with DMA Avalon-ST with SR-IOV	选择连接到应用层的接口。
Hard IP mode	Gen3x8, Interface: 256-bit, 250 MHz Gen3x4, Interface: 256-bit, 125 MHz Gen3x4, Interface: 128-bit, 250 MHz Gen3x2, Interface: 128-bit, 125 MHz Gen3x2, Interface: 64-bit, 250 MHz Gen3x1, Interface: 64-bit, 125 MHz Gen2x8, Interface: 256-bit, 125 MHz Gen2x8, Interface: 128-bit, 250 MHz Gen2x4, Interface: 128-bit, 125 MHz Gen2x2, Interface: 64-bit, 125 MHz Gen2x4, Interface: 64-bit, 250 MHz Gen2x1, Interface: 64-bit, 125 MHz Gen1x8, Interface: 128-bit, 125 MHz Gen1x8, Interface: 64-bit, 250 MHz Gen1x4, Interface: 64-bit, 125 MHz Gen1x2, Interface: 64-bit, 125 MHz Gen1x1, Interface: 64-bit, 125 MHz Gen1x1, Interface: 64-bit, 62.5 MHz	选择以下元件： <ul style="list-style-type: none"> lane 数据速率。支持 Gen1, Gen2, 和 Gen3 FPGA 架构中硬核 IP 事务层与已实现应用层之间数据接口的宽度 应用层接口频率 Avalon-ST with SR-IOV 接口仅支持 256-bit 模式。
Port type	Native Endpoint Root Port	指定端口类型。 端点将参数储存于 Type 0 配置空间。根端口将参数储存于 Type 1 配置空间。 根端口不支持带有 DMA 的 Avalon-MM 接口。 Avalon-ST with SR-IOV 接口仅支持 Native Endpoint 操作。
RX Buffer credit allocation - performance for received requests	Minimum Low Balanced High Maximum	决定 posted 头 (header) 信用, posted 数据信用, non-posted 头信用, 完成头信用, 和 16 KB RX 缓冲器中完成数据信用的分配。可通过此设置调节信用分配从而优化您的系统。 所选设置信用分配显示于 Message 窗口中。 Message 窗口根据您选择地改变而动态更新 Posted, Non-Posted Header 和数据, 及 Completion Headers 和数据的信用数。 请参阅 吞吐量优化 章节, 了解更多关于优化您设计的信息。 请参阅以下 基于接口类型的可用 RX 缓冲器分配选择 内容, 了解这些基于接口类型的设计的可用性。

继续...

参数	值	说明
		<p>Minimum—以 PCIe 规范允许的最小量配置 non-posted 和 posted 请求信用，并保留大部分的 RX 缓冲器空间用于已接收的完成头（completion header）及数据。为应用逻辑生成许多读请求，但仅极少接收 PCIe 链路单个请求的 variation 选择该选项。</p> <p>Low—为 non-posted 和 posted 请求信用配置稍微大量的 RX 缓冲空间，但仍把大部分空间专用于已接收完成头和数据。为应用逻辑生成许多读请求，但仅极少接收 PCIe 链路单个请求的 variation 选择该选项。建议为通用端点应用（位于端点应用层逻辑中 DMA 引擎生成大部分 PCIe 数据流量）选择该项。</p> <p>Balanced—配置大约一半的 RX 缓冲器空间给已接收请求，另一半 RX 缓冲器空间配置给已接收完成。为已接收请求和已接收完成基本一样多的 variation 选择此选项。</p> <p>High—为已接收请求配置大部分的 RX 缓冲器空间并分配稍大于最小量的空间给已接收完成。当大部分 PCIe 请求由 PCIe 链路的另一端生成且局部应用层逻辑仅偶尔生成少量读请求时，选择该项。建议此选项用于一般根端点应用（由位于端点的 DMA 引擎在该一般根端点应用中生成大部分 PCIe 数据流量）。</p> <p>Maximum—以 PCIe 规范允许的最小量配置完成空间，保留大部分 RX 缓冲空间给已接收请求。当大部分 PCIe 请求由 PCIe 链路的另一端生成且本地应用层逻辑从不或仅偶尔生成少量读请求时，选择该项。建议此选项用于控制和状态端点应用（控制和状态端点应用自身不生成 PCIe 请求，而只作为根复合体读和写请求的目标）。</p>
RX Buffer completion credits	Header credits, Data credits	显示从信用分配参数得出的 16KB RX 缓冲器中完成信用的数量。每个头信用为 16 bytes。每个数据信用为 20 bytes。

相关链接

PCI Express Base Specification 3.0

3.1.1. 基于接口类型的 RX 缓冲分配可用选择

表 13. 基于接口类型的 RX 缓冲分配可用选择

接口类型	Minimum	Low	Balanced	High	Maximum
Avalon-ST	可用	可用	可用	可用	可用
Avalon-MM	可用	可用	可用	不可用	不可用
支持 DMA 的 Avalon-MM	可用	可用	可用	不可用	不可用
支持 SE-IOV 的 Avalon-ST	可用	可用	可用	可用	可用

3.2. Arria 10 Avalon-MM 设置

表 14. Avalon Memory-Mapped（存储器映射）系统设置

参数	值	说明
Avalon-MM address width	32-bit 64-bit	为访问 Avalon 地址域中 Avalon-MM 从端口的 Avalon-MM RX 主端口指定地址宽度。当选择 32-bit 地址时，PCI Express Avalon-MM 桥执行地址转换。指定 64-bit 地址时，两个方向中都不进行地址转换。将指定目的地址原封不动转发到 Avalon-MM 接口。

继续...



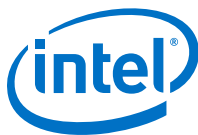
参数	值	说明
		支持 DMA 的 Avalon-MM 接口，该值必须设置为 64 。
Enable control register access (CRA) Avalon-MM slave port	On/Off	允许通过指定从端口对互连架构中桥寄存器的读和写访问。 Requester/Completer variant 要求选择此项，但对于 Completer Only variant 为可选。使能该项可进行桥寄存器读和写访问，Completer-Only 单 dword variation 除外。
Export MSI/MSI-X conduit interfaces	On/Off	此项设置为 On ，内核导出可用于实现 MSI 和 MSI-X 中断自定义处理的顶层 MSI 和 MSI-X 接口。请参阅 <i>Interrupts for End Points Using the Avalon-MM Interface with Multiple MSI/MSI-X Support</i> ，了解更多关于自定义中断处理的信息。如果此项设置为 Off ，内核内部处理中断。
Enable hard IP status bus when using the Avalon-MM interface	On/Off	此项设置为 On 时，您的顶层 variant 包含有助于调试的信号，如，链路训练和状态，以及错误信号。顶层 variant 包含下列信号： <ul style="list-style-type: none"> 链路状态信号 ECC 错误信号 LTSSM 信号 配置奇偶校验位错误信号
Instantiate Internal Descriptor Controller	On/Off	此项设置为 On 时，描述符控制器被包含于 Avalon-MM DMA 桥中。关闭此选项时，其中的描述符控制器会作为一个单独的外部组件。如要在设计中使用 Intel 提供的描述符控制器，就开启此项。如要修改或替换设计中的描述符控制器逻辑，请关闭此项。
Enable burst capabilities for RXM BAR2 port	On/Off	设置此项为 On 时，BAR2 RX Avalon-MM 主端口具有突发性。如果 BAR2 为 32 bits 可突发，则 BAR3 不可作为其他用途。如果 BAR2 为 64 bits，BAR3 寄存器保留地址的上层 32 bits。
Enable 256 tags	On/Off	此项设置为 On 时，内核支持 256 个标签，改善高延迟系统的性能。打开 Control 寄存器中的 Extended Tagbit 后，在此开启该项。
Address width of accessible PCIe memory space	20-64	指定必要的位数以访问 PCIe 地址空间。

3.3. 基地址寄存器（BAR）设置

端口类型决定 BAR 的可用类型和大小。

表 15. BAR 寄存器

参数	值	说明
Type	Disabled 64-bit prefetchable memory 32-bit non-prefetchable memory 32-bit prefetchable memory I/O address space	<p>如果选择 64-bit 预可取存储器，两个连续的 BAR 就被合并组成一个 64-bit 预可取 BAR；必须将编号较高的 BAR 设置为 Disabled。因为一般系统中不支持 64-bit BAR，所以根端口 Type 1 配置空间将非预可取存储器设置为 32 bits。也可把各 BAR 配置成单独的 32-bit 存储器。</p> <p>将存储器定义为预可取以允许预取连续数据。当请求程序（requestor）从同样的区域中要求比原本需求更多的数据时，预可取存储器是有优势的。如果要指定某个存储器为预可取，则它必须具备以下两个属性：</p> <ul style="list-style-type: none"> 读取时不会产生例如更改已读取数据值的副作用 允许合并写入 <p>32-bit prefetchable memory 和 I/O address space BAR 仅适用于 Legacy Endpoint。</p>
Size	N/A	连接您的组件后，Qsys 自动计算所需要的大小。



3.4. 器件识别寄存器

表 16. 器件 ID 寄存器

以下表格列出了只读器件 ID 寄存器的默认值。可使用参数编辑器更改这些寄存器的值。请参阅 *Type 0 Configuration Space Registers* 了解关于 Device Identification registers（器件识别寄存器）的布局。

寄存器名称	范围	默认值	说明
Vendor ID	16-bit	0x00001172	设置 Vendor ID 寄存器的只读值。根据 <i>PCI Express</i> 规范，此参数不可设置为 0xFFFF。 地址偏移：0x000。
Device ID	16-bit	0x0000e001	设置 Device ID 寄存器的只读值。该寄存器只在 Type 0（端点）配置空间中有效。 地址偏移：0x000。
Revision ID	8-bit	0x00000000	设置 Revision ID 寄存器的只读值。 地址偏移：0x008。
Class code	24-bit	0x00000000	设置 Class Code 寄存器的只读值。 地址偏移：0x008。
Subsystem Vendor ID	16-bit	0x00000000	设置 PCI Type 0 配置空间中 Subsystem Vendor ID 寄存器的只读值。根据 <i>PCI Express Base</i> 规范此参数不可设置到 0xFFFF。该值由 PCI-SIG 分配给器件制造商。此寄存器只在 Type 0（端点）配置空间中有效。 地址偏移：0x02C。
Subsystem Device ID	16-bit	0x00000000	设置 PCI Type 0 配置空间中的 Subsystem Device ID 寄存器只读值。 地址偏移：0x02C。

[相关链接](#)

[PCI Express Base Specification 3.0](#)

3.5. PCI Express 和 PCI 性能参数

此参数组定义 IP 内核的各种性能的属性。其中一些参数存储于 PCI 配置空间—PCI 可兼容的配置空间。字节偏移表示参数地址。

3.5.1. 器件性能

表 17. 性能寄存器

参数	可能的值	默认值	说明
Maximum payload size	128 bytes 256 bytes 512 bytes 1024 bytes 2048 bytes	128 bytes	指定所支持的最大有效负载量。此参数设置器件性能寄存器（0x084[2:0]）所支持的最大有效负载容量域的只读值。地址偏移：0x084。 用于 Avalon-MM 接口和 Avalon-MM with DMA 接口的 Maximum payload size （最大有效负载）是 256 Byte。
Completion timeout range	ABCD BCD ABC AB B A	ABCD	表示用于可选的完成超时可编程性机制的器件功能支持。该机制允许系统软件修改完成超时的值。该域只适用于发布自身请求的根端口及端点。对于 Avalon-MM with DMA 接口，该参数必须设置为 NONE 。在 <i>PCI</i>
继续...			



参数	可能的值	默认值	说明
	None		<p><i>Express Capability Structure Version</i> 的 Device Control 2 register (0x0A8) 中指定和使能完成超时。由该域保留的所有其他功能，必须被硬接线到 0x0000b。已定义四个时间值范围如下：</p> <ul style="list-style-type: none"> 范围 A: 50 us-10 ms 范围 B: 10 ms-250 ms 范围 C: 250 ms-4 s 范围 D: 4 s-64 s <p>设置位以显示所支持的超时值范围。该功能实现的超时值必须在 50 s 到 50 ms 范围内。以下值指定的范围为：</p> <ul style="list-style-type: none"> None—不支持完成超时编程 0001 范围 A 0010 范围 B 0011 范围 A 和 B 0110 范围 B 和 C 0111 范围 A, B, 和 C 1110 范围 B, C 和 D 1111 范围 A, B, C, 和 D <p>保留所有其他值。Intel 建议完成超时机制的失效时间长于 10 ms。</p>
Disable completion timeout	On/Off	On	禁用完成超时机制。选择 On 时，内核支持 via PCI Express Device Control Register 2 的完成超时禁用机制。应用层逻辑必须根据要求的范围实现实际的完成超时机制。

3.5.2. 错误报告

表 18. 错误报告

参数	值	默认值	说明
Advanced error reporting (AER)	On/Off	Off	当 On 时，使能 Advanced Error Reporting (AER) (高级报错) 功能。
ECRC checking	On/Off	Off	当 On 时，使能 ECRC 查看。将 Advanced Error Capabilities and Control Register (高级报错功能与控制寄存器) 中可 ECRC 查看的位设置为只读。此参数要求使能 AER 功能。
ECRC generation	On/Off	Off	当 On 时，使能 ECRC 生成功能。将 Advanced Error Capabilities and Control Register (高级报错功能与控制寄存器) 中可 ECRC 生成的位设置为只读。此参数需要使能 AER 功能。
Enable ECRC forwarding on the Avalon-ST interface	On/Off	Off	当为 On 时，使能 ECRC 转发到应用层。Avalon-ST RX 路径上，接收 TLP 包括 ECRC dword ⁽¹⁾ ，并设置 TD 位，如果存在 ECRC。发送来自应用层的 TLP 必须包括 ECRC dword 及已设置的 TD 位。
Track RX completion buffer overflow on the Avalon-ST interface	On/Off	Off	当为 On 时，内核包含用于追踪 RX 已发送的完成高速缓冲上溢状态的 rxfx_cplbuf_ovf 输出状态信号。

相关链接

[PCI Express Base Specification Revision 3.0](#)



3.5.3. 链路性能

表 19. 链路性能

参数	值	说明
Link port number (仅根端口)	0x01	设置 Link Capabilities 寄存器中端口数量域的只读值。此参数仅用于根端口。并应保持不变。
Data link layer active reporting (Root Port only)	On/Off	把根端口的该参数设置为 On ，如果附带的端点支持报告数据链路控制的 DL_Active 状态和状态机管理的可选功能。对于可热插播的端点（如 Slot Capabilities 寄存器 Hot Plug Capable 域所显示），该参数必须为 On 。对于不支持此可选功能的根端口组件，请将该选项设置为 Off 。 不可用于 Avalon-MM 或 Avalon-MM DMA 接口。
Surprise down reporting (Root Port only)	On/Off	当此选项设置为 On 时，端点支持检测和报告意外瘫痪错误情况的可选功能。从根端口读取该错误情况。 不可用于 Avalon-MM 或 Avalon-MM DMA 接口。
Slot clock configuration	On/Off	此选项设置为 On 时，表示端点 或根端口 使用的物理参考时钟与系统在连接器上提供的物理参考时钟相同。设置为 Off 时，IP 内核使用独立的时钟，即使连接器上已有一个参考时钟。此参数用于设置 PCI Express Link Status 寄存器中 Slot Clock 配置位 (bit 12)。

3.5.4. MSI 和 MSI-X Capabilities

表 20. MSI 和 MSI-X Capabilities

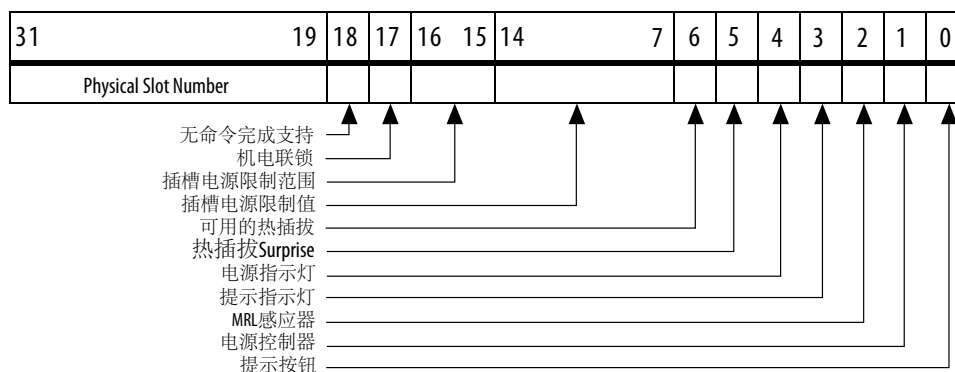
参数	值	说明
MSI messages requested	1、2、4、8、16、32	指定应用层可请求的消息数。设置 Message Control 寄存器的 Multiple Message Capable 域值，Address: 0x050[31:16]。
MSI-X Capabilities		
Implement MSI-X	On/Off	当为 On 时，添加 MSI-X 功能性。
	Bit Range	
Table size	[15:0]	系统软件读取该域以决定 MSI-X Table 的大小 $\langle n \rangle$ ，其编码为 $\langle n-1 \rangle$ 。例如，返回值 2047 表示 table 大小为 2048。该域为只读。合法范围是 0-2047 (2^{16})。 Address offset: 0x068[26:16]
Table Offset	[31:0]	指向 MSI-X Table 底部。列表 BAR 指示器 (BIR) 中较低的 3 个位被软件设置为 0，以组成一个 64-bit qword 对齐的偏移。该域为只读。
Table BAR Indicator	[2:0]	从配置空间的 0x10 开端出指定函数的一个 BAR 以将 MSI-X Table 映射到存储器空间。该域为只读。合法范围是 0-5。
Pending Bit Array (PBA) Offset	[31:0]	用作其中一个函数基地址寄存器中的地址的偏移并指向 MSI-X PBA 的底部。PBA BIR 中较低的 3 个位被软件设置为 0 从而组成一个 32-bit qword 对齐的偏移。此域为只读。
Pending BAR Indicator	[2:0]	从配置空间中 0x10 开端处指定函数基地址寄存器，从而将 MSI-X PBA 映射到存储器空间。该域为只读。合法范围 0-5。
注释： 1. 本用户指南中的术语 word，dword 和 qword 与 PCI Express Base Specification 中的意义相同。一个 word 是 16-bit，一个 dword 是 32-bit，以及一个 qword 是 64-bit。		

3.5.5. 插槽性能

表 21. 插槽性能

参数	值	说明
Use Slot register	On/Off	如果端口上有实现的插槽，则该插槽需要被用于根端口。插槽状态被记录在 PCI Express Capabilities 寄存器中。仅根端口模式支持该参数。 定义插槽特性。选择 Enable slot capability 开启该项。请参考以下图示了解位的定义。 不可用于 Avalon-MM DMA。
Slot power scale	0 - 3	在 Slot power limit 中指定已使用的比例。定义为以下系数： <ul style="list-style-type: none"> 0 = 1.0x 1 = 0.1x 2 = 0.01x 3 = 0.001x 硬件和固件初始化之前的默认值是 b' 00。写入此寄存器时会导致端口发送 Set_Slot_Power_Limit 消息。 请参阅 <i>PCI Express Base Specification Revision</i> 中 6.9 小节了解更多信息。
Slot power limit	0 - 255	结合 Slot power scale value ，指定插槽供电电源的瓦数上限。更多信息请参阅 <i>PCI Express Base Specification</i> 7.8.9 小节。
Slot number	0-8191	指定插槽个数

图 5. 插槽性能



3.5.6. 电源管理

表 22. 电源管理参数

参数	值	说明
Endpoint L0s acceptable latency	Maximum of 64 ns Maximum of 128 ns Maximum of 256 ns Maximum of 512 ns Maximum of 1 us Maximum of 2 us Maximum of 4 us	此设计参数指定器件与根端复合体之间任意链路退出 L0s 状态时，器件可接受的最长延迟。它设置 Device Capabilities Register (0x084) 中端点 L0s 可接受延迟域的只读值。 此端点不支持 L0s 或 L1 状态。然而，已切换过的系统中可能存在链路与已使能 L0s 和 L1 的切换连接。设置此参数，系统配置软件读取系统中所有器件的可接受延迟，以及每个链路的退出延迟从而决定用于使能 Active State Power Management (ASPM) 的链路。该设置对根端口禁用。

继续...



参数	值	说明
	No limit	此参数的默认值为 64 ns，也是大部分设计的最安全设置。
Endpoint L1 acceptable latency	Maximum of 1 us Maximum of 2 us Maximum of 4 us Maximum of 8 us Maximum of 16 us Maximum of 32 us No limit	该值表示从 L1 到 L0 状态的转换中端点能承受的可接受延迟。它也是端点内部高速缓冲的间接测量。它设置 Device Capabilities Register 中端点 L1 可接受延迟域为只读。 此端点不支持 L0s 或 L1 状态。但切换过的系统中可能有链路已被连接到使能的 L0s 和 L1 切换。设置此参数以允许系统配置软件读取系统中所有器件的可接受延迟，以及每个链路的退出延迟 (exit latency) 从而决定用于使能 Active State Power Management (ASPM) 的链路。该设置对根端口禁用。 此参数的默认值是 1 μ s，也是大部分设计的最安全设置。

3.5.7. 供应商指定扩展性能 (VSEC)

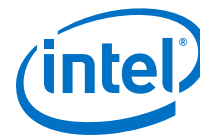
表 23. VSEC

参数	值	说明
Vendor Specific Extended Capability (VSEC) ID:	0x00001172	设置供应商指定扩展功能中的 16-bit 用户 ID 寄存器的只读值。
Vendor Specific Extended Capability (VSEC) Revision:	0x00000000	设置供应商指定扩展功能中的 4-bit 用户 ID 寄存器的只读值。
User Device or Board Type ID register from the Vendor Specific Extended Capability:	0x00000000	设置供应商指定扩展功能中的 16-bit 器件或电路板类型 ID 寄存器的只读值。

3.6. 配置，调试，和扩展项

表 24. PCI Express 的系统设置

参数	值	说明
Enable configuration via Protocol (CvP)	On/Off	设置为 On 时，Quartus Prime 软件将端点放置于 configuration via protocol (CvP) 所要求的位置。点击以下 Configuration via Protocol (CvP) 的链接了解更多关于 CvP 的信息。
Enable dynamic reconfiguration of PCIe read-only registers	On/Off	设置为 On 时，可以使用硬核 IP 重配置总线动态重配置硬核 IP 只读寄存器。更多信息请参阅 Hard IP Reconfiguration Interface 。
Enable transceiver dynamic reconfiguration	On/Off	设置为 On 时，创建一个可由软件驱动的 Avalon-MM 从接口以用于收发器寄存器更新。
Enable Altera Debug Master Endpoint (ADME)	On/Off	设置为 On 时，嵌入的 Altera Debug Master Endpoint (Altera 调试主端点) 内部链接到 Avalon-MM 从接口以进行动态重配置。ADME 可访问收发器中的重配置空间。它使用 JTAG 通过 System Console 运行测试和调试功能。
Enable Arria 10 FPGA Development Kit connection	On/Off	设置为 On 时，添加控制和状态管道接口到顶层 variant，并被连接到 PCIe 开发套件的一个组件。



3.7. PHY 特性

表 25. PHY 特性

参数	值	说明
Gen2 TX de-emphasis	3.5dB 6dB	指定 Gen2 的发送去加重。Intel 建议遵循以下设置： <ul style="list-style-type: none"> 3.5dB: 短 PCB 走线 6.0dB: 长 PCB 走线
Requested equalization far-end TX preset	Preset0-Preset9	为 Phase 2 和 3 远端发送器指定所请求的 TX 预设。默认值 Preset8 为大多数设计提供最好的信号质量。

3.8. Arria10 设计实例

表 26. 设计实例

参数	值	说明
Available Example Designs	DMA PIO	选择 DMA 项时，所生成的设计包含一个直接的存储器访问应用。该应用包含上游和下游传输。当选择 PIO 项时，所生成的设计包含一个仅可下游传输的目标应用。DMA 设计实例不可用于 Quartus Prime Pro - Stratix 10 Edition Beta 发布。
Simulation	On/Off	设置为 On 时，所生成的输出包含一个仿真模型。
Synthesis	On/Off	设置为 On 时，所生成的输出包含一个综合模型。综合不适用于 Quartus Prime Pro - Stratix 10 Edition Beta 发布。
Generated HDL format	Verilog	仅支持 Verilog HDL。
Target Development Kit	Arria 10 GX FPGA Development Kit Arria 10 GX FPGA Development Kit ES2 None	选择 Arria 10 FPGA Development Kit 用于 Arria 10 生产器件。选择 Arria 10 FPGA Development Kit ES 用于工程采样 (ES) 或 ES2 器件。选择 None ，如果您自己开发板为对象。

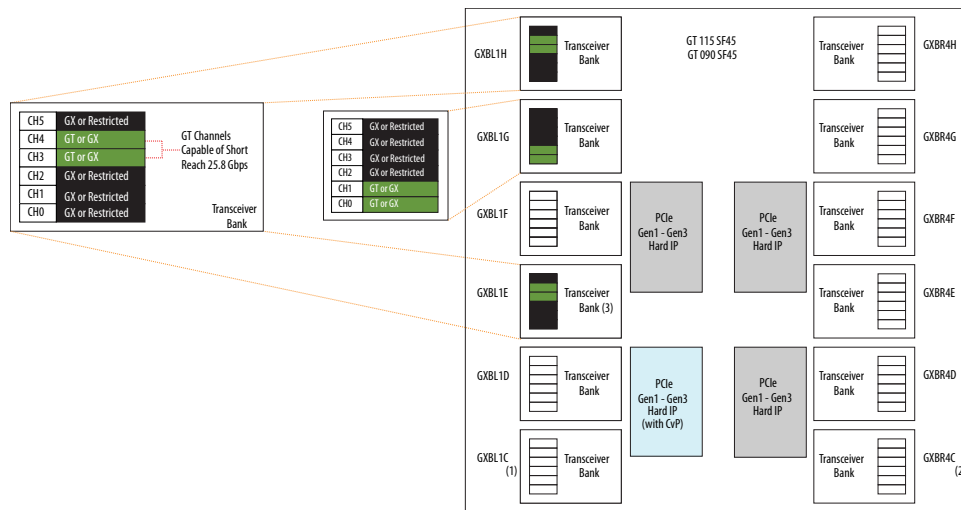
4. Arria 10 器件中 Hard IP 的物理布局

Arria 10 器件包括用于 PCI Express 的 1 到 4 个硬核 IP 模块。底部左边的硬核 IP 模块具有用于覆晶封装的 (Flip Chip packages) 的 CvP 功能性。在其他封装类型中, CvP 功能性位于底部右边模块。

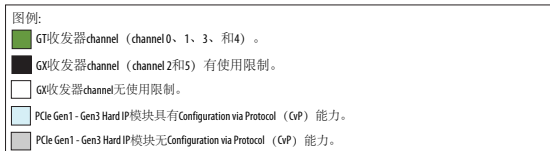
注意:

Arria 10 器件不支持使用 Gen3 x4 或 Gen3 x8 IP 内核配置底部 (左边或右边) 硬核 IP 块, 以及使用 Gen3 x1 或 Gen3 x2 IP 内核 variation 配置顶部同侧硬核 IP 块的配置。

图 6. Arria 10 器件具有 72 个收发器 Channel 和 4 个 PCIe Hard IP 块



注释:
(1) 左列底部收发器组始终以“C”结尾命名。
(2) 右列底部收发器组会以“C”、“D”、或“E”结尾命名。
(3) 如果GT channel用于收发器组GXBL1E, 则无法使用与GXBL1F和GXBL1E相邻的PCIe HIP。



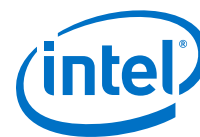
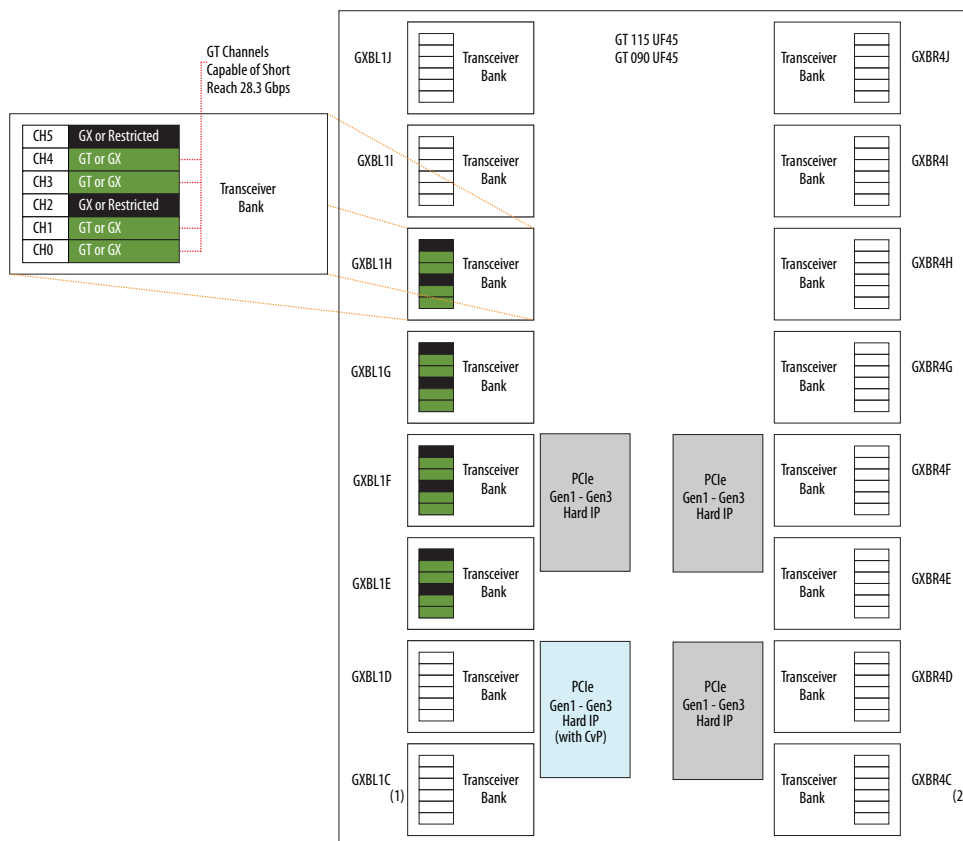


图 7. Arria 10 器件具有 96 个收发器 Channel 和 4 个 PCIe Hard IP 块



注释:
(1) 左列底部收发器组始终以“C”结尾命名。
(2) 右列底部收发器组会以“C”、“D”、或“E”结尾命名。

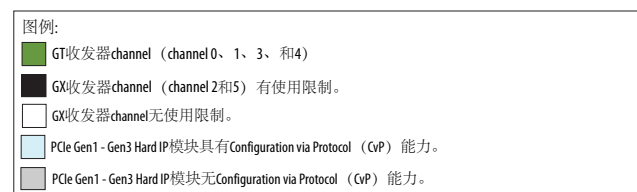
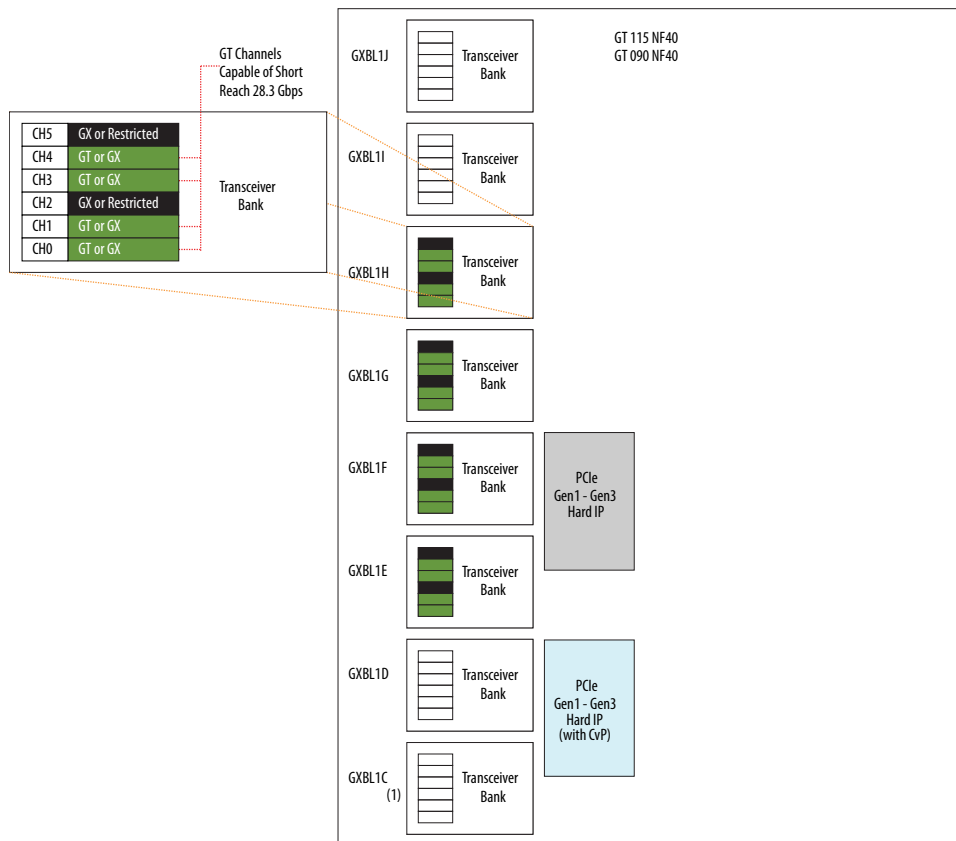


图 8. Arria 10 GT 器件具有 48 个收发器 Channel 和 2 个 PCIe Hard IP 块



注释:
(1)左列底部收发器组始终以“C”结尾命名。
(2)这些器件中的收发器仅位于器件左侧。

图例:

- GT收发器channel (channel 0、1、3、和14)。
- GX收发器channel (channel 2和5) 有使用限制。
- GX收发器channel无使用限制。
- PCIe Gen3 HIP模块具有Configuration via Protocol (CvP) 能力。
- PCIe Gen3 HIP模块无Configuration via Protocol (CvP) 能力。

请参阅 *Intel FPGA Arria 10 Transceiver PHY User Guide* 中的 *Arria 10 Transceiver Layout* 部分了解关于 Arria 10 GT、GX、和 SX 器件的全面数据。

相关链接

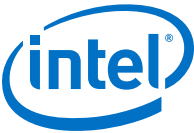
[Intel FPGA Arria 10 Transceiver PHY IP Core 用户指南](#)

更多关于收发器物理 (PHY) 层体系结构, PLLs, 时钟网络, 和收发器 PHY IP 的信息。

4.1. Gen1, Gen2, Gen3 数据速率的 Channel 和管脚布局

下图说明用于 Arria 10 Hard IP for PCI Express 的 x1、x2、x4、及 x8 channel 和管脚布局。

图示中, 没有被用于 PCI Express 协议的 Channel 可用于其他协议。未使用 Channel 为灰色。



注意: 所有配置中，PCS 中的物理 channe 4 连接到硬核 IP 中的逻辑 channe 0。您不能改变下图所示的 channe 布局。

请参阅本章开头显示各类 Arria 10 器件中硬核 IP PCIe 模块物理位置的图示，获得 <txvr_block_N>和<txvr_block_N+1>的可能值。对于每 HIP 模块而言，收发器为 <txvr_block_N>，且与其相并在其下方扩展。位于<txvr_block_N>正上方的收发器模块是 <txvr_block_N+1>。例如，在配备 96 个收发器通道和 4 个 PCIe HIP 模块的 Arria 器件中，如果您的设计使用支持 CvP 的 HIP 模块，则 <txvr_block_N>为 GXB1C，以及<txvr_block_N+1>为 GXB1D。

图 9. Arria 10 Gen1, Gen2, Gen3 x1 Channel 和管脚布局

<txvr_block_N>_TX/RX_CH4N	PMA Channel 5	PCS Channel 5	Hard IP for PCIe
	PMA Channel 4	PCS Channel 4	
	PMA Channel 3	PCS Channel 3	
	PMA Channel 2	PCS Channel 2	
	PMA Channel 1	PCS Channel 1	
	PMA Channel 0	PCS Channel 0	
	PMA Channel 5	PCS Channel 5	
	PMA Channel 4	PCS Channel 4	Hard IP Ch0
	PMA Channel 3	PCS Channel 3	
	PMA Channel 2	PCS Channel 2	
	PMA Channel 1	PCS Channel 1	
	PMA Channel 0	PCS Channel 0	

图 10. Arria 10 Gen1, Gen2, Gen3 x2 Channel 和管脚布局

<txvr_block_N>_TX/RX_CH5N <txvr_block_N>_TX/RX_CH4N	PMA Channel 5	PCS Channel 5	Hard IP for PCIe
	PMA Channel 4	PCS Channel 4	
	PMA Channel 3	PCS Channel 3	
	PMA Channel 2	PCS Channel 2	
	PMA Channel 1	PCS Channel 1	
	PMA Channel 0	PCS Channel 0	
	PMA Channel 5	PCS Channel 5	
	PMA Channel 4	PCS Channel 4	Hard IP Ch0
	PMA Channel 3	PCS Channel 3	
	PMA Channel 2	PCS Channel 2	
	PMA Channel 1	PCS Channel 1	
	PMA Channel 0	PCS Channel 0	

图 11. Arria 10 Gen1, Gen2, Gen3 x4 Channel 和管脚布局

	PMA Channel 5	PCS Channel 5	Hard IP for PCIe
	PMA Channel 4	PCS Channel 4	
	PMA Channel 3	PCS Channel 3	
	PMA Channel 2	PCS Channel 2	
<txvr_block_N+1>_TX/RX_CH1N	PMA Channel 1	PCS Channel 1	
<txvr_block_N+1>_TX/RX_CH0N	PMA Channel 0	PCS Channel 0	
<txvr_block_N>_TX/RX_CH5N	PMA Channel 5	PCS Channel 5	Hard IP Ch0
<txvr_block_N>_TX/RX_CH4N	PMA Channel 4	PCS Channel 4	
	PMA Channel 3	PCS Channel 3	
	PMA Channel 2	PCS Channel 2	
	PMA Channel 1	PCS Channel 1	
	PMA Channel 0	PCS Channel 0	

图 12. Arria 10 Gen1, Gen2, Gen3 x8 Channel 和管脚布局

<txvr_block_N+1>_TX/RX_CH5N	PMA Channel 5	PCS Channel 5	Hard IP for PCIe
<txvr_block_N+1>_TX/RX_CH4N	PMA Channel 4	PCS Channel 4	
<txvr_block_N+1>_TX/RX_CH3N	PMA Channel 3	PCS Channel 3	
<txvr_block_N+1>_TX/RX_CH2N	PMA Channel 2	PCS Channel 2	
<txvr_block_N+1>_TX/RX_CH1N	PMA Channel 1	PCS Channel 1	
<txvr_block_N+1>_TX/RX_CH0N	PMA Channel 0	PCS Channel 0	
<txvr_block_N>_TX/RX_CH5N	PMA Channel 5	PCS Channel 5	Hard IP Ch0
<txvr_block_N>_TX/RX_CH4N	PMA Channel 4	PCS Channel 4	
	PMA Channel 3	PCS Channel 3	
	PMA Channel 2	PCS Channel 2	
	PMA Channel 1	PCS Channel 1	
	PMA Channel 0	PCS Channel 0	

4.2. Gen1 和 Gen2 数据速率的 Channel 布局 and fPLL 使用

下图说明用于 Arria 10 Hard IP for PCI ExpressT 的 x1, x2, x4, 和 x8 channel 布局。图示中，未被用于 PCI Express 协议的 channel 可用于其他协议。未使用的通道为灰色。

注意: 所有配置中， PCS 中的物理通道 4 连接到硬核 IP 中的逻辑通道 0。您不能改变下图所示的通道布局。



图 13. Arria 10Gen1 和 Gen2 x1 Channel 布局

fPLL1	PMA Channel 5	PCS Channel 5	Hard IP for PCIe
ATX1 PLL	PMA Channel 4	PCS Channel 4	
	PMA Channel 3	PCS Channel 3	
fPLLO	PMA Channel 2	PCS Channel 2	
ATX0 PLL	PMA Channel 1	PCS Channel 1	
	PMA Channel 0	PCS Channel 0	Hard IP Ch0
fPLL1 <small>Master CGB</small>	PMA Channel 5	PCS Channel 5	
ATX1 PLL	PMA Channel 4	PCS Channel 4	
	PMA Channel 3	PCS Channel 3	
fPLLO	PMA Channel 2	PCS Channel 2	
ATX0 PLL	PMA Channel 1	PCS Channel 1	
	PMA Channel 0	PCS Channel 0	

图 14. Arria 10Gen1 和 Gen2 x2 Channel 布局

fPLL1	PMA Channel 5	PCS Channel 5	Hard IP for PCIe
ATX1 PLL	PMA Channel 4	PCS Channel 4	
	PMA Channel 3	PCS Channel 3	
fPLLO	PMA Channel 2	PCS Channel 2	
ATX0 PLL	PMA Channel 1	PCS Channel 1	
	PMA Channel 0	PCS Channel 0	Hard IP Ch0
fPLL1 <small>Master CGB</small>	PMA Channel 5	PCS Channel 5	
ATX1 PLL	PMA Channel 4	PCS Channel 4	
	PMA Channel 3	PCS Channel 3	
fPLLO	PMA Channel 2	PCS Channel 2	
ATX0 PLL	PMA Channel 1	PCS Channel 1	
	PMA Channel 0	PCS Channel 0	

图 15. Arria 10Gen1 和 Gen2 x4 Channel 布局

fPLL1	PMA Channel 5	PCS Channel 5	Hard IP for PCIe
ATX1 PLL	PMA Channel 4	PCS Channel 4	
	PMA Channel 3	PCS Channel 3	
fPLLO <small>Master CGB</small>	PMA Channel 2	PCS Channel 2	
ATX0 PLL	PMA Channel 1	PCS Channel 1	
	PMA Channel 0	PCS Channel 0	Hard IP Ch0
fPLL1	PMA Channel 5	PCS Channel 5	
ATX1 PLL	PMA Channel 4	PCS Channel 4	
	PMA Channel 3	PCS Channel 3	
fPLLO	PMA Channel 2	PCS Channel 2	
ATX0 PLL	PMA Channel 1	PCS Channel 1	
	PMA Channel 0	PCS Channel 0	

图 16. Gen1 和 Gen2 x8 通道布局

fPLL1	PMA Channel 5	PCS Channel 5	Hard IP for PCIe
ATX1 PLL	PMA Channel 4	PCS Channel 4	
	PMA Channel 3	PCS Channel 3	
fPLLO <small>Master CGB</small>	PMA Channel 2	PCS Channel 2	
ATX0 PLL	PMA Channel 1	PCS Channel 1	
	PMA Channel 0	PCS Channel 0	
fPLL1	PMA Channel 5	PCS Channel 5	Hard IP Ch0
ATX1 PLL	PMA Channel 4	PCS Channel 4	
	PMA Channel 3	PCS Channel 3	
fPLLO	PMA Channel 2	PCS Channel 2	
ATX0 PLL	PMA Channel 1	PCS Channel 1	
	PMA Channel 0	PCS Channel 0	

4.3. Gen3 数据速率的 Channel 布局，以及 fPLL 和 ATX PLL 的使用

下图说明用于 Arria 10 Hard IP for PCI Express 的 x1, x2, x4, 和 x8 channel 布局。

Gen3 variant 必须以 Gen1 数据速率进行初始训练。因此，Gen3 variant 需要 fPLL 来生成 2.5 和 5.0 Gbps 块，以及 ATX PLL 来生成 8.0 Gbps 块。这些图示中，未用于 PCI Express 协议的通道可用于其他协议。未使用通道为灰色。

注意: 所有配置中，PCS 中的物理通道 4 连接到硬核 IP 中的逻辑通道 0。您不能改变下图所示的通道布局。

图 17. Arria 10 Gen3 x1 Channel 布局

fPLL1	PMA Channel 5	PCS Channel 5	Hard IP for PCIe
ATX1 PLL	PMA Channel 4	PCS Channel 4	
	PMA Channel 3	PCS Channel 3	
fPLLO	PMA Channel 2	PCS Channel 2	
ATX0 PLL	PMA Channel 1	PCS Channel 1	
	PMA Channel 0	PCS Channel 0	
fPLL1	PMA Channel 5	PCS Channel 5	Hard IP Ch0
ATX1 PLL <small>Master CGB</small>	PMA Channel 4	PCS Channel 4	
	PMA Channel 3	PCS Channel 3	
fPLLO	PMA Channel 2	PCS Channel 2	
ATX0 PLL	PMA Channel 1	PCS Channel 1	
	PMA Channel 0	PCS Channel 0	



图 18. Arria 10 Gen3 x2 Channel 布局

fPLL1	PMA Channel 5	PCS Channel 5	Hard IP for PCIe
ATX1 PLL	PMA Channel 4	PCS Channel 4	
	PMA Channel 3	PCS Channel 3	
fPLL0	PMA Channel 2	PCS Channel 2	
ATX0 PLL	PMA Channel 1	PCS Channel 1	
	PMA Channel 0	PCS Channel 0	
fPLL1	PMA Channel 5	PCS Channel 5	Hard IP Ch0
ATX1 PLL <small>Master CGB</small>	PMA Channel 4	PCS Channel 4	
	PMA Channel 3	PCS Channel 3	
fPLL0	PMA Channel 2	PCS Channel 2	
ATX0 PLL	PMA Channel 1	PCS Channel 1	
	PMA Channel 0	PCS Channel 0	

图 19. Arria 10 Gen3 x4 Channel 布局

fPLL1	PMA Channel 5	PCS Channel 5	Hard IP for PCIe
ATX1 PLL	PMA Channel 4	PCS Channel 4	
	PMA Channel 3	PCS Channel 3	
fPLL0	PMA Channel 2	PCS Channel 2	
ATX0 PLL <small>Master CGB</small>	PMA Channel 1	PCS Channel 1	
	PMA Channel 0	PCS Channel 0	
fPLL1	PMA Channel 5	PCS Channel 5	Hard IP Ch0
ATX1 PLL	PMA Channel 4	PCS Channel 4	
	PMA Channel 3	PCS Channel 3	
fPLL0	PMA Channel 2	PCS Channel 2	
ATX0 PLL	PMA Channel 1	PCS Channel 1	
	PMA Channel 0	PCS Channel 0	

图 20. Gen3 x8 Channel 布局

fPLL1	PMA Channel 5	PCS Channel 5	Hard IP for PCIe
ATX1 PLL	PMA Channel 4	PCS Channel 4	
	PMA Channel 3	PCS Channel 3	
fPLL0	PMA Channel 2	PCS Channel 2	
ATX0 PLL <small>Master CGB</small>	PMA Channel 1	PCS Channel 1	
	PMA Channel 0	PCS Channel 0	
fPLL1	PMA Channel 5	PCS Channel 5	Hard IP Ch0
ATX1 PLL	PMA Channel 4	PCS Channel 4	
	PMA Channel 3	PCS Channel 3	
fPLL0	PMA Channel 2	PCS Channel 2	
ATX0 PLL	PMA Channel 1	PCS Channel 1	
	PMA Channel 0	PCS Channel 0	

4.4. PCI Express Gen3 Bank 组使用限制

与具 Gen3 能力的有效 PCI Express 接口共享 bank 组的收发器 channel 都有下列限制。包括 Hard IP 和 Soft IP 实现：

- VCCR_GXB 和 VCCT_GXB 设置为 1.03 V 或 1.12 V 时，bank 组中非 PCIe channel 可用于 chip-to-chip 应用的最大数据速率为 12.5 Gbps。这些 channel 不可用于驱动背板或 GT 速率。
- VCCR_GXB 和 VCCT_GXB 设置为 0.95 V 时，这些 bank 组中的非 PCIe channel 不可用。

仅可用 Gen1 和 Gen2 的 PCI Express 接口不受影响。

状态

影响下列 ES 器件：

- Arria 10 10AX660 ES2 and 10AX480 ES 器件
- Arria 10 10AX115/10AX090 ES3 器件
- Arria 10 GX 660 ES2 and Arria 10 GX 480 ES 器件

无修复计划。



5. IP 内核接口

本章节介绍使用 Avalon-MM DMA 接口的 Arria 10 Hard IP for PCI Express 的顶层信号。Avalon-MM DMA 桥包括高性能，可突发 Read DMA 和 Write DMA 模块。控制 Read DMA 和 Write DMA 模块的 DMA 描述符控制器可包含于 Avalon-MM DMA 桥或被单独例化。它使用 64-bit 寻址，从而无需地址转换。单独例化的描述符控制器管理 Read DMA 和 Write DMA 模块。此 variant 可用于下列配置：

- Gen1 x8
- Gen2 x4
- Gen2 x8
- Gen3 x2
- Gen3 x4
- Gen3 x8

5.1. Arria10 DMA Avalon-MM DMA 接口到应用层

本小节介绍当 PCIe variant 中包含高性能，可突发 Read DMA 和 Write DMA 模块时，其顶层接口的情况。

取决于器件的不同，到应用层的接口可为 128 或 256-bit。

图 21. 带有内部描述符控制器的 Avalon-MM DMA 桥

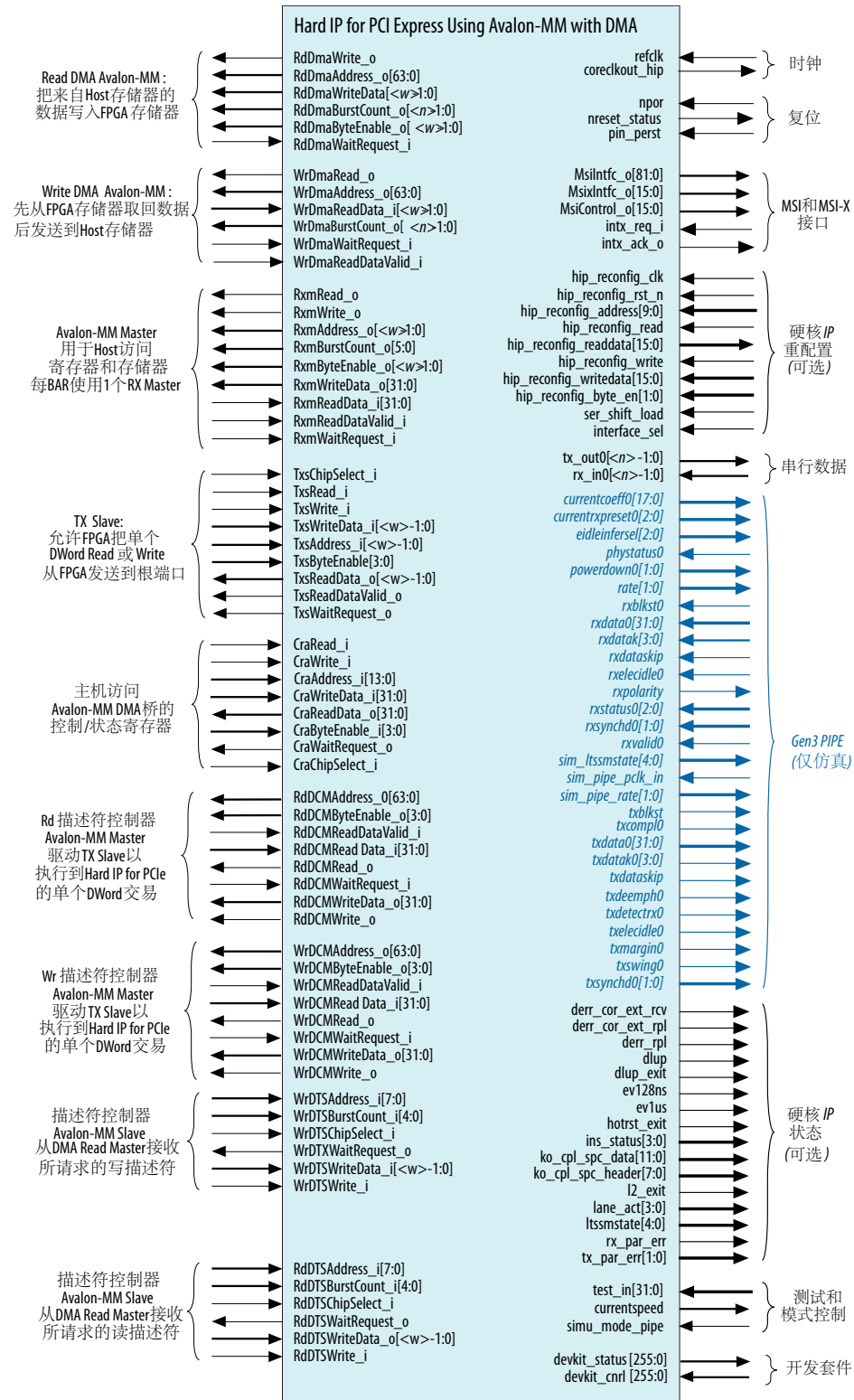
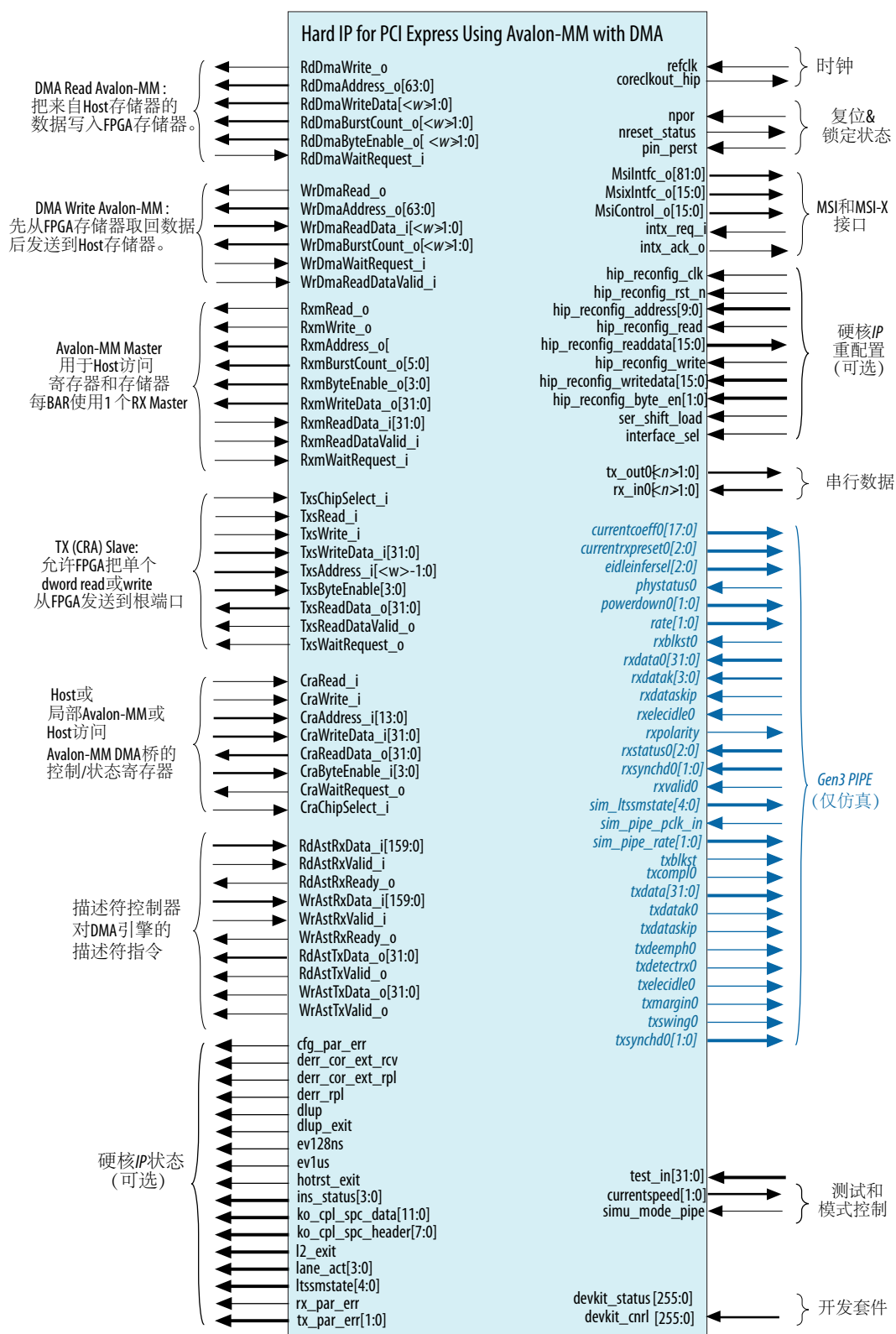


图 22. 带有外部描述符控制器的 Avalon-MM DMA 桥



本小节介绍需要实现 DMA 的接口。其他所有接口的介绍在下个小节，*Avalon-MM 接口到应用层* 中。

5.1.1. 读 DMA Avalon-MM 主端口

Read DMA 模块发送存储器读 TLP 上游。并把写完成数据通过高吞吐量读主端口写入外部 Avalon-MM 接口。此端口处理 IP 内核从 DMA 描述符控制器接收的描述符。

Read DMA Avalon-MM Master Port 接口运行两个功能：

1. 为描述符控制器提供描述符列表

此模块发送存储器读请求以从使用 Avalon-ST 读接口上游存储器的主存储器中取回描述符列表。此模块通过 Avalon-MM 接口把描述符条目写入描述符控制器 FIFO。

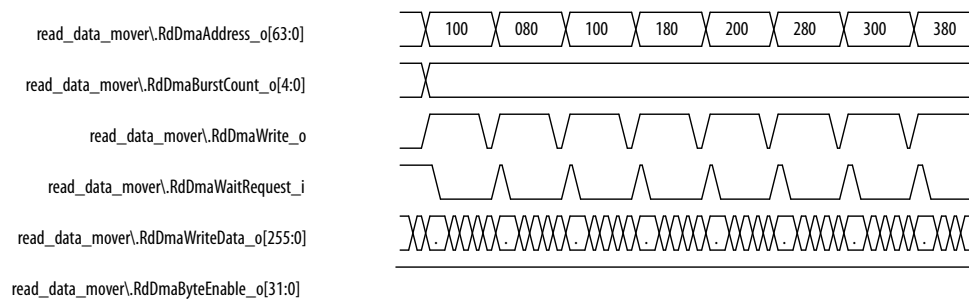
2. 写数据到 Avalon-MM 空间的存储器中

在 DMA 读通过普通 DMA-Read 操作完成从主存储器源地址取回数据后，读 DMA 模块通过此接口把数据写入 Avalon-MM 地址空间中的目的地址。

表 27. Read DMA 256-Bit Avalon-MM Master 接口

信号名称	方向	说明
RdDmaWrite_o	输出	置位后，表示读 DMA 模块已预备把读完成数据写入 Avalon-MM 地址空间中的存储器组件。
RdDmaAddress_o[63:0]	输出	在 Avalon-MM 地址空间中，为读完成数据指定写入地址。
RdDmaWriteData_o[127 or 255:0]	输出	要被写入 Avalon-MM 地址空间的都完成数据。
RdDmaBurstCount_o[4:0] or [5:0]	输出	以 128-或 256-bit 字指定突发计数。此总线中的 5-bit 用于 256-bit 接口。6-bit 用于 128-bit 接口。
RdDmaByteEnable_o[15 or 31:0]	输出	指定字(word)中的有效字节。
RdDmaWaitRequest_i	输入	置位后，表示存储器还未就绪接收数据。

图 23. Read DMA Avalon-MM Master 写数据到 FPGA 存储器



5.1.2. 写 DMA Avalon-MM 主端口

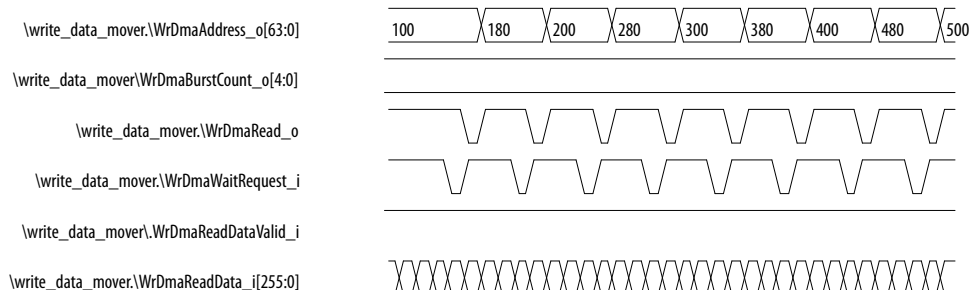
在发布向主存储器传输数据的存储器写请求前，写 DMA 模块通过该接口从 Avalon-MM 地址空间取回数据。



表 28. DMA Read 256-Bit Avalon-MM Master 接口

信号名称	方向	说明
WrDMARead_o	输出	置位后，表示写 DMA 模块从 Avalon-MM 地址空间的存储器组件读取数据以写入 PCIe 地址空间。
WrDmaAddress_o[63:0]	输出	在 Avalon-MM 地址空间中为从存储器组件中读取的数据指定地址。
WrDmaReadData_i[127 or 255:0]	输入	指定 Write DMA 模块将要写入 PCIe 地址空间的完成数据。
WrDmaBurstCount_o[4:0] or [5:0]	输出	以 128-或 256-bit 字指定突发计数。此总线中的 5 位用于 256-bit 接口，6 位用于 128-bit 接口。
WrDmaWaitRequest_i	输入	置位后，表示存储器还未就绪被读取。
WrDmaReadDataValid_i	输入	置位后，表示 WrDmaReadData_i 有效

图 24. Write DMA Avalon-MM Master 从 FPGA 存储器读取数据



5.1.3. RX Master 模块

RX Master 模块把从 PCIe 链路接收到的读和写 TLP 转换成 Avalon-MM 请求，并用于连接到互连中的 Qsys 组件。此模块允许其他 PCIe 组件，包括主机软件，访问连接到 Qsys 系统中的其他 Avalon-MM 从端口。

如未使能突发模式，RX Master 模块仅支持 32-bit 读或写请求。如果未使能突发模式，RX Master 模块仅支持 32-bit 读或写请求。从 PCIe 链路收到的所有其他请求都被视为此器件编程模式违规，从而被作为 PCIe Completer Abort 状态处理。可使用 32-bit 寻址使能 BAR2 突发模式，或使用 64-bit 寻址使能 BAR2 和 BAR3 突发模式。使能后，此模式支持双字，突发读，或写请求。当内部例化描述符控制器时，用于 BAR0 的 RX 主端口被内部使用并不可用于其他用途。

表 29. 用于 BAR 访问的 RX 主控制接口端口

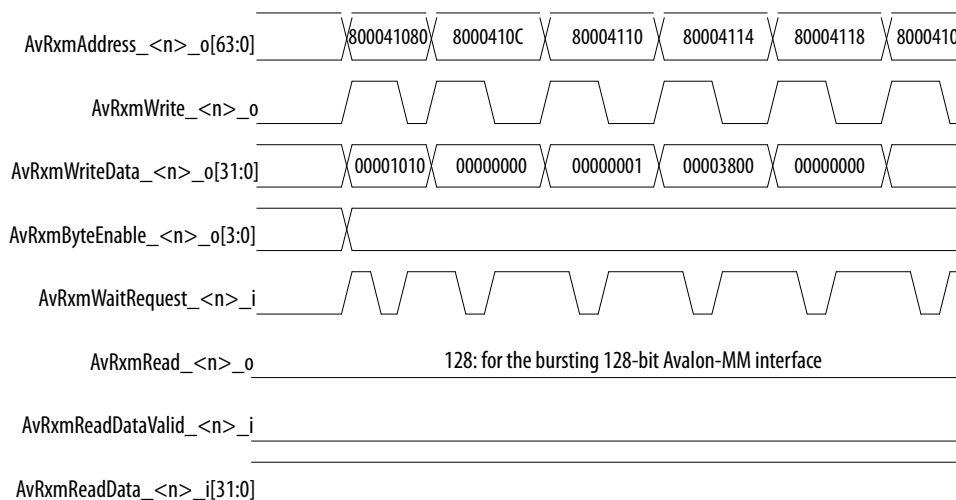
每个 BAR 都有一个相应的 RX 主控制接口。下列表格中，<n>是 BAR 号码。

信号名称	方向	说明
RxmRead_<n>_o	输出	置位后，表示一个 Avalon-MM 读请求。
RxmWrite_<n>_o	输出	置位后，表示一个 Avalon-MM 写请求。
RxmAddress_<n>_o[<w>-1:0]	输出	指定 Avalon-MM 字节地址。因为所有地址都是字节地址，此地址中有涵义的位是 [<w>-1:2]。Bits 1 和 0 的值有一个为 0。<w>可以为 32 或 64。
RxmBurstCount_<n>_o[5:0]	输出	以 dwords (32 bits) 指定突发计数。开启 Enable burst capabilities for RXM BAR2 ports 时，此可选信号可用于 BAR2。

继续...

信号名称	方向	说明
RxmByteEnable_<n>_o[<w>:0]	输出	为将要被写入的数据指定有效字节。<w>值如下： <ul style="list-style-type: none"> 4：表示非突发的 RX Master 32：表示突发 128-bit Avalon-MM 接口 64：表示突发 256-bit Avalon-MM 接口
RxmDataWrite_<n>_o[<w>:0]	输出	指定 Avalon-MM 写数据。<w>值如下： <ul style="list-style-type: none"> 32：表示非突发 RX Master 128：表示突发 128-bit Avalon-MM 接口 256：表示突发 256-bit Avalon-MM 接口
RxmReadData_<n>_i[<w>:0]	输入	指定 Avalon-MM 读数据。<w>值如下： <ul style="list-style-type: none"> 32：表示非突发 RX Master 128：表示突发 128-bit Avalon-MM 接口 256：表示突发 256-bit Avalon-MM 接口
RxmReadDataValid_<n>_i[31:0]	输入	置位后，表示 RxmReadData_i[31:0]有效。
RxmWaitRequest_<n>_i	输入	置位后表示控制寄存器访问 Avalon-MM 从端口还未准备响应。

图 25. RXM Master 写入 Avalon-MM 地址空间中的存储器



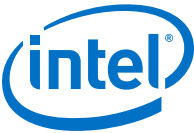
5.1.4. TX Slave 模块

TX Slave 模块将 Avalon-MM master 读和写请求转换成用于根端口的 PCI Express TLP。

TX Slave 控制模块支持单个未处理非突发请求。它通常发送状态更新到主机。这是一个 32-bit Avalon-MM 从总线。

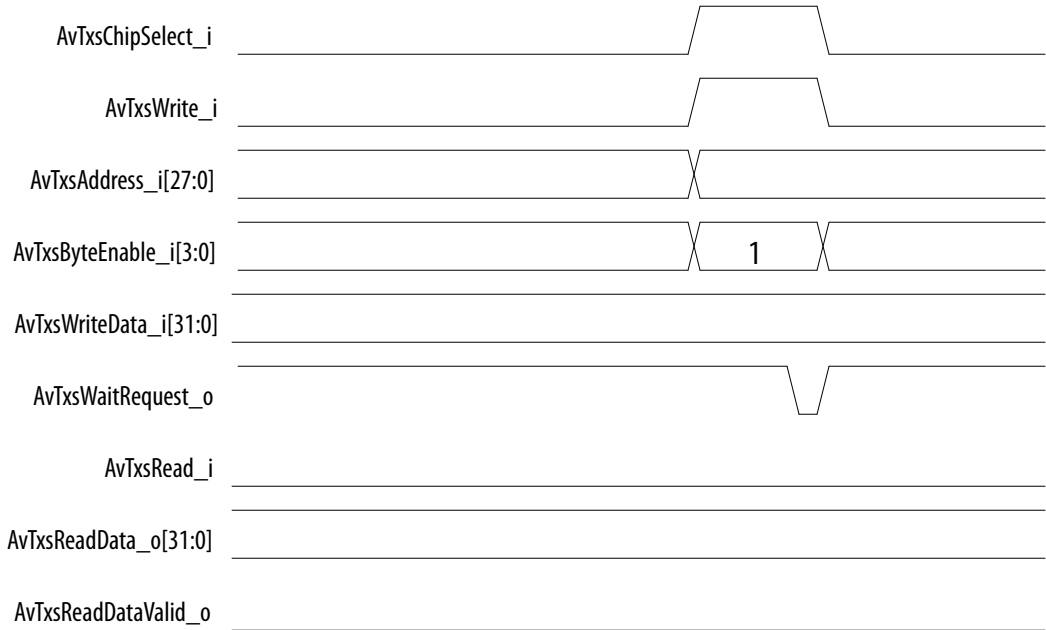
表 30. TX Slave 控制

信号名称	方向	说明
TxsChipSelect_i	输入	置位后，表示已选择该从接口。仅当已置位读和写信号时，才必须置位该信号。直到有效读数据已返回才需要置位该信号。
TxsRead_i	输入	置位后，从根复合体或根端口指定一个 TX Avalon-MM slave 读请求。
继续...		



信号名称	方向	说明
TxsWrite_i txs_write_i	输入	置位后，为根复合体或根端口指定一个 TX Avalon-MM slave 写请求。
TxsWriteData_i[31:0]	输入	指定写命令的 Avalon-MM 数据。
TxsAddress_i[<w>-1:0]	输入	指定读和写命令的 Avalon-MM 字节地址。通过参数 Address width of accessible PCIe memory space 指定该地址总线的宽度。
TxsByteEnable_i[3:0]	输入	指定写命令的有效字节。
TxsReadData_o[31:0]	输出	指定读完成数据。
TxsReadDataValid_o	输出	置位后，表示读数据有效。
TxsWaitRequest_o	输出	置位后，表示 Avalon-MM 从端口还未准备响应读或写请求。

图 26. TX Slave 接口发送状态到主机



5.1.5. 32-Bit 非突发 Avalon-MM 控制寄存器访问（CRA）Slave 信号

CRA 端口为 Avalon-MM DMA 桥的状态寄存器提供只读主机访问。

表 31. Avalon-MM CRA 从接口信号

信号名称	方向	说明
CraRead_i	输入	读使能。 当前版本的 CRA 从接口为只读。将此信号归于 Avalon-MM 接口部分，以致在未来还可进行改进。
CraWrite_i	输入	写请求。

继续...

信号名称	方向	说明
CraAddress_i[13:0]	输入	为控制寄存器分配 16 KB 的地址空间。Avalon-MM slave 地址提供下至从数据总线宽度的地址解析。因为所有地址都是字节地址，所以该地址逻辑上降至 bit 2。Bits 1 和 0 为 0。使用字节使能来进行 dword 的独立字节读和写。例如，要写字节 0 和 1，将该信号设置成 4'b0011。
CraWriteData_i[31:0]	输入	写数据。当前版本的 CRA 从接口为只读。将此信号归于 Avalon-MM 接口部分，以致在未来还可进行改进。
CraReadData_o[31:0]	输出	读数据线路 (lines)
CraByteEnable_i[3:0]	输入	字节使能。
CraWaitRequest_o	输出	等待请求以推迟额外请求。
CraChipSelect_i	输入	到该从端口的片选信号。
CraIrq_o	输出	中断请求。端口的 Avalon-MM 中断请求。

相关链接

DMA 描述符控制器寄存器 (第 70 页)

5.1.6. Avalon-ST 描述符控制接口单独例化时

从主机存储器的描述符列表中取回多个描述符条目后，描述符控制器组成单个 160-bit 描述符指令并把它发送到读 DMA 或写 DMA 引擎。

表 32. 从描述符控制器到读 DMA 引擎的描述符指令接口

信号名称	方向	说明
RdAstRxData_i[159:0]	输入	为 Read DMA 模块指定描述符。请参阅下列 <i>DMA Descriptor Format</i> 列表了解关于位的定义。
RdAstRxValid_i	输入	置位后，表示数据有效。
RdAstRxReady_o	输出	置位后，表示 Read DMA 读模块已准备接收新的描述符。就绪延迟为 3 个周期。因此，就绪被置位后，接口可接受数据 3 个周期。

表 33. 从描述符控制器到写 DMA 引擎的描述符指令接口

信号名称	方向	说明
WrAstRxData_i[159:0]	输入	为 Write DMA 模块指定描述符。请参阅 <i>DMA Descriptor Format</i> 列表了解关于位的定义。
WrAstRxValid_i	输入	置位后，表示数据有效。
WrAstRxReady_o	输出	置位后，表示 Write DMA 模块引擎已准备接收新的描述符。该信号的就绪延迟为 3 个周期。因此，就绪被置位后，接口可接受数据 3 个周期。

5.1.6.1. Avalon-ST 描述符状态接口

当 DMA 模块完成了一个描述符指令处理时，就通过下列接口将 DMA 状态返回到描述符控制器。

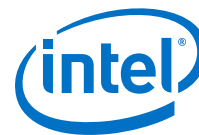


表 34. 从读 DMA 引擎到描述符控制器的读 DMA 状态接口

信号名称	方向	说明
RdAstTxData_o[31:0]	输出	驱动状态信息到描述符控制器组件。请参阅下列 <i>DMA Status Bus</i> 列表了解更多信息。
RdAstTxValid_o	输出	置位后，表示数据有效。

表 35. 从 Write DMA 引擎到描述符控制器的 Write DMA 状态接口

信号名称	方向	说明
WrAstTxData_o[31:0]	输出	驱动状态信息到描述符控制器组件。请参阅下列 <i>DMA Status Bus</i> 列表了解更多关于该总线的信息。
WrAstTxValid_o	输出	置位后，表示数据有效。

表 36. DMA 描述符格式

Bits	名称	说明
[31:0]	Source Low Address	DMA 源地址的低阶 32 bits。此地址边界必须 32 bits 对齐，从而 2 个最次重要位的值为 2'b00。对于读 DMA 模块，源地址是 PCIe 域地址。而写 DMA 模块的源地址是 Avalon-MM 域地址。
[63:32]	Source High Address	源地址的高阶 32 bits。
[95:64]	Destination Low Address	DMA 目的地址的低阶 32 bits。此地址边界必须 32 bits 对齐，从而 2 个最低有效位的值为 2'b00。对于读 DMA 模块，目的地址是 Avalon-MM 域地址。而写 DMA 模块的目的地址是 PCIe 域地址。
[127:96]	Destination High Address	目的地址的高阶 32 位。
[145:128]	DMA Length	以 dwords 指定 DMA 长度。此长度必须大于 0。最大长度为 1 MB 到 4 bytes。
[153:146]	DMA Descriptor ID	最多指定 128 个描述符。
[158:154]	Reserved	—
[159]	Immediate Write	设置为 1'b1 时，Write DMA 引擎运行一次 Immediate Write。Immediate Write 提供发送 Write TLP 上游的快速机制。描述符存储 32-bit 负载，以替代描述符的 Source Low Address 域。

5.1.6.2. DMA 描述符状态总线

描述符成功完成时，Read DMA 和 Write DMA 模块向 RdDmaTxData_o[31:0] 或 WrDmaTxData_o[31:0] 总线上的描述符控制器报告状态。

以下表格表示 DMA 描述符状态总线上触发事件的映射：

表 37. DMA 状态总线

Bits	名称	说明
[31:9]	Reserved.	—
[8]	Done	置位后，单个 DMA 描述符已成功完成。
[7:0]	Descriptor ID	其状态已被报告的描述符 ID。

5.1.7. 描述符控制器接口内部例化时

5.1.7.1. 读描述符控制器 Avalon-MM 主端口

读描述符控制器 Avalon-MM 主端口驱动 TX Avalon-MM 从端口。该端口将单个 dword 交易驱动到 Arria10 Avalon-MM DMA for PCIe。读描述符控制器使用此端口写描述符状态到 PCIe 域且当 MSI 消息被使能时很可能写描述符状态到 MSI。该 Avalon-MM 主端口仅可用于内部例化的描述符控制器。

表 38. 读描述符控制器 Avalon-MM 主端口

信号名称	方向	说明
RdDCMAddress_o[63:0]	输出	为读数据指定地址。
RdDCMByteEnable_o[3:0]	输出	指定数据的有效字节。
RdDCMReadDataValid_i	输入	置位时，表示读数据有效。
RdDCMReadData_i[31:0]	输入	接收来自单个 dword TX 从接口的读数据。
RdDCMRead_o	输出	置位后，表示读交易。
RdDCMWaitRequest_i	输入	置位后，表示 Avalon-MM slave 器件还未准备响应。
RdDCMWriteData_o[31:0]	输出	驱动单个 dword 写到已连接的从端口。
RdDCMWrite_o	输出	置位后，表示写交易。

5.1.7.2. 写描述符控制器 Avalon-MM 主端口

表 39. 写描述符控制器 Avalon-MM 主端口

Avalon-MM 描述符控制器主接口是一个支持等待请求的 32-bit 单 dword 主接口。写描述符控制器使用此端口将状态写回到 PCI-Express 域并且当 MSI 被使能时很可能写回到 MSI。该 Avalon-MM 主端口仅可用于内部例化的描述符控制器。

信号名称	方向	说明
WrDCMAddress_o[63:0]	输出	为写数据指定地址。
WrDCMByteEnable_o[3:0]	输出	指定数据的有效字节。
WrDCMReadDataValid_i	输入	置位后，表示读数据有效。
WrRdDCMReadData_i[31:0]	输出	从单个 dword TX 从接口接收即将被写入的数据。
WrDCMRead_o	输出	置位后，表示读交易。
WrDCMWaitRequest_i	输入	置位后，表示 Avalon-MM 从器件还未准备响应。
WrDCMWriteData_o[31:0]	输出	驱动单个 dword 写数据。
WrDCMWrite_o	输出	置位后，表示写交易。

5.1.7.3. 读描述符列表 Avalon-MM 从端口

当您选择内部描述符控制器时，此端口可用。它接收由 Read DMA 取回的 Read DMA 描述符。将此端口连接到读 DMA Avalon-MM 主端口。



表 40. 读描述符列表 Avalon-MM 从端口

信号名称	方向	说明
RdDTSAddress_i[7:0]	输入	指定描述符地址。
RdDTSBurstCount_i[4:0] or [5:0]	输入	指定突发计数字。
RdDTSChipSelect_i	输入	置位后，表示针对该从端口的读。
RdDTSWriteData_i[255:0] or [127:0]	输入	驱动写数据。
RdDTSWrite_i	输入	置位后，表示写交易。
RdDTSWaitRequest_o	输出	置位后，表示 Avalon-MM 从器件还未准备响应。

5.1.7.4. 写描述符列表 Avalon-MM 从端口

当您选择内部描述符控制器时，此端口可用。它接收由 Read DMA 取回的 Write DMA 描述符。将此端口连接到 Read DMA Avalon-MM 主端口。

表 41. 写描述符列表 Avalon-MM 从端口

信号名称	方向	说明
WrDTSAddress_i[7:0]	输入	为写数据指定描述符地址。
wr_dts_burst_count_i[4:0] or [5:0]	输入	以字为单位指定突发计数。
WrDTSChipSelect_i	输入	置位后，表示用于此从端口的写。
WrDTSWaitRequest_o	输出	置位后，表示 Avalon-MM 从器件还未准备响应。
WrDTSWriteData_i[255:0] or [127:0]	输入	驱动写数据。
WrDTSWrite_i	输入	置位后，表示写交易。

5.2. 时钟信号

表 42. 时钟信号

信号	方向	说明
refclk	输入	用于 IP 内核的参考时钟。它的频率必须符合参数编辑器中 System Settings 标题下的指定频率。它是专用 REFCLK 管脚的专用自由运行输入时钟。
coreclkout_hip	输出	这是数据链路和事务层使用的固定频率时钟。

[相关链接](#)

[时钟 \(第 85 页\)](#)

5.3. 复位，状态，和链路训练信号

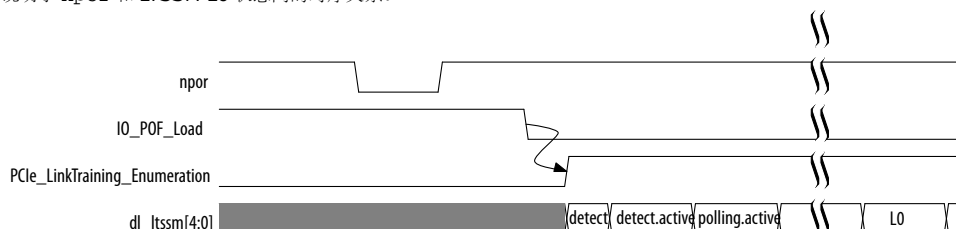
请参阅 *Reset and Clocks* 了解更多关于复位序列和复位逻辑框图的信息。

表 43. 复位信号

信号	方向	说明
npor	输入	有效低电平复位信号。在 Intel 硬件设计实例中，npor 是软件应用层 pin_perst 和 local_rstn 的 OR。如果您未驱动来自于应用层的软复位信号，那么必须从 pin_perst。得到此信号。无法禁用此信号。复位整个 IP 内核和收发器。异步。此信号为 <i>edge</i> (边沿)， <i>not level</i> 敏感 (无电平敏感)；因此，不能在此信号中使用低电平值来保持复位中的自定义逻辑。更多关于复位控制器的信息，请参阅 <i>Reset</i> 。
nreset_status	输出	有效低电平复位信号。产生于 npor 或 pin_perstn。可使用此信号复位应用层。
pin_perst	输入	器件 PCIe 复位管脚的有效低电平。pin_perst 复位数据通路和控制寄存器。Configuration via Protocol (CvP)需要该信号。更多关于 CvP 的信息，请参阅 <i>Arria 10 CvP Initialization and over PCI Express User Guide</i> 。 Arria10 器件最多可有 4 个 Hard IP for PCI Express 实例 (instance)。每个实例都有自己的 pin_perst 信号。必须把每个 Hard IP 实例的 pin_perst 连接到器件中相应的 nPERST 管脚。这些管脚在下列位置： <ul style="list-style-type: none"> • NPERSTL0: 左下 Hard IP 和 CvP 块 • NPERSTL1: 左上 Hard IP 块 • NPERSTR0: 右下 Hard IP 块 • NPERSTR1: 右上 Hard IP 块 例如，如果您正在使用器件左下角的 Hard IP 实例，就必须把 pin_perst 连接到 NPERSTL0。 为了最大限度使用 Arria10 器件，Intel 建议首先使用左下的 Hard IP。它是唯一一个通过 PCIe 链路支持 CvP 的位置。如果您的设计不需要 CvP，就可选择其他 Hard IP 块。 请参阅 <i>Arria 10 GX, GT, and SX Device Family Pin Connection Guidelines</i> 了解更多关于这些管脚的信息。

图 27. 复位和链路训练时序关系

下图说明了 npor 和 LTSSM L0 状态间的时序关系。



注意: 为了满足 100 ms 的系统配置时间，必须通过支持 CvP 的快速被动并行配置方案以及一个 32-bit 数据宽度 (FPPx32) 或使用自主模式中的 Arria10 Hard IP for PCI Express。

表 44. 状态和链路训练信号

信号	方向	说明
cfg_par_err	输出	表示 TLP 中的一个奇偶校验错误布线到内部配置空间。此错误还被记录在供应商指定的扩展功能内部错误寄存器中。如果出现此错误您必须复位 Hard IP。
derr_cor_ext_rcv	输出	表示 RX 缓冲中的一个已更正错误。此信号仅用于调试。直到 RX 缓冲中存有数据则此信号有效。这是脉冲信号而不是电平信号。内部上来讲，此脉冲由 500 MHz 时钟生成。脉冲扩展卡将信号扩展，从而运行于 250 MHz 的 FPGA 架构可采集到它。因为此错误由 IP 内核更正，所以不需要应用层干扰。 ⁽²⁾
derr_cor_ext_rpl	输出	表示重试缓冲中的已更正 ECC 错误。此信号仅用于调试。因为此错误由 IP 内核更正，所以不需要应用层干扰。 ⁽²⁾
继续...		



信号	方向	说明
derr_rpl	输出	表示重试缓冲中的不可更正错误。此信号仅用于调试。 ⁽²⁾
dlup	输出	置位后，表示 Hard IP 块处于数据链路控制和管理状态机(DLCMSM) DL_Up 状态。
dlup_exit	输出	IP 内核退出 DLCMSM DL_Up 状态时，此信号被置位一个 pld_clk 周期，表示数据链路层已失去与 PCIe 链路另一端的通信并离开上电状态。当此脉冲被置位，应用层应该生成一个内部复位信号，且被置位至少 32 个周期。
evl28ns	输出	置位每 128 ns，从而创建时间基准对齐活动。
evlus	输出	置位每 1μs，从而创建时间基准对齐活动。
hotrst_exit	输出	热复位退出。当 LTSSM 退出热复位状态时，此信号被置位一个时钟周期。此信号会导致应用层复位。此信号为有效低电平。当此脉冲被置位，应用层应该生成一个内部复位信号，且此信号被置位至少 32 个周期。
int_status[3:0]	输出	这些信号把传统中断 (legacy interrupt) 驱动到应用层，详情如下： <ul style="list-style-type: none"> int_status[0]: 中断信号 A int_status[1]: 中断信号 B int_status[2]: 中断信号 C int_status[3]: 中断信号 D
ko_cpl_spc_data[11:0]	输出	应用层可使用该信号建立电路以阻止完成数据的 RX 高速缓冲器上溢。端点必须为完成数据播发无限空间；然而，RX 高速缓冲器空间是有限的。ko_cpl_spc_data 是一个静态信号，它反映出可存储于完成 RX 高速缓冲器的 16 字节完成数据单元的总数。
ko_cpl_spc_header[7:0]	输出	应用层可使用该信号建立电路以阻止完成头 (header) 的 RX 高速缓冲器上溢。端点必须为完成头 (header) 播发无限空间；然而，RX 高速缓冲器空间是有限的。ko_cpl_spc_header 是一个静态信号，它显示出可存储于 RX 高速缓冲器的完成头 (header) 的总数。
l2_exit	输出	L2 退出。该信号为有效低电平，否则依然保持高电平。在来自于 l2.idle 的 LTSSM 发送开始检测后，该信号被置位一个周期 (值从 1 更改到 0，然后改回到 1)。当该脉冲被置位，应用层将生成一个内部复位信号，并置位该生成信号至少 32 个周期。
lane_act[3:0]	输出	Lane 有效模式：该信号显示链路训练期间执行配置的通道数。编码定义如下： <ul style="list-style-type: none"> 4' b0001: 1 个 lane 4' b0010: 2 个 lane 4' b0100: 4 个 lane 4' b1000: 8 个 lane
ltssmstate[4:0]	输出	LTSSM 状态：LTSSM 状态机编码定义以下状态： <ul style="list-style-type: none"> 00000: Detect.Quiet (检测。静音) 00001: Detect.Active (检测。有效) 00010: Polling.Active (轮询。有效) 00011: Polling.Compliance (轮询。合规) 00100: Polling.Configuration (轮询。配置) 00101: Polling.Speed (轮询。速度) 00110: Config.Linkwidthstart (配置。链路宽度开始) 00111: Config.Linkaccept (配置。链路接受) 01000: Config.Lanenumaccept (配置。通道数接受) 01001: Config.Lanenumwait (配置。通道数等待) 01010: Config.Complete (配置。完成) 01011: Config.Idle (配置。空闲)

继续...

⁽²⁾ Intel 不会严格地测试和验证调试信号。只是使用调试信号来观察行为。请不要使用调试信号驱动自定义逻辑。

信号	方向	说明
		<ul style="list-style-type: none"> 01100: Recovery.Rcvlock (恢复。恢复时钟) 01101: Recovery.Rcvconfig (恢复。恢复配置) 01110: Recovery.Idle (恢复。空闲) 01111: L0 10000: Disable (禁用) 10001: Loopback.Entry (环回。进入) 10010: Loopback.Active (环回。有效) 10011: Loopback.Exit (环回。退出) 10100: Hot.Reset (热。复位) 10101: L0s 11001: L2.transmit.Wake (L2 发送。唤醒) 11010: Speed.Recovery (速度。恢复) 11011: Recovery.Equalization, Phase 0 (恢复。均衡, Phase 0) 11100: Recovery.Equalization, Phase 1 (恢复。均衡, Phase 1) 11101: Recovery.Equalization, Phase 2 (恢复。均衡, Phase 2) 11110: Recovery.Equalization, Phase 3 (恢复。均衡, Phase 3) 11111: Recovery.Equalization, Done (恢复。均衡, 完成)
rx_par_err	输出	<p>置位单个周期, 表示在 RX 缓冲输入的 TLP 中检测到一个奇偶校验错误。该错误在 VSEC 寄存器中被记录为无法更正的内部错误。请参阅 <i>Uncorrectable Internal Error Status Register</i> (无法更正的内部错误状态寄存器) 了解更多信息。如果出现该错误您必须复位硬核 IP, 因为奇偶校验错误会将 Hard IP 保持在未知状态。</p>
tx_par_err[1:0]	输出	<p>置位单个周期, 表示在 TX TLP 发送期间有一个奇偶校验错误。这些错误被记录在 VSEC 寄存器中。编码定义如下:</p> <ul style="list-style-type: none"> 2' b10: TX 事务层检测到奇偶校验错误。TLP 无效且在 VSEC 寄存器中记录为无法更正的内部错误。请参阅 <i>Uncorrectable Internal Error Status Register</i> (无法更正的内部错误状态寄存器) 了解更多信息。 2' b01: 稍后, TX 数据链路层检测到奇偶错误, 并驱动 2' b01 显示该错误。当检测到该错误时复位 IP 内核。如果复位失败, 请联络 Intel 技术支持。 <p>注释: 不是所有的仿真模型把事务层错误位与数据链路层错误位协同置位。</p>

相关链接

- [Arria 10 CvP Initialization and Partial Reconfiguration over PCI Express User Guide](#)

5.4. 端点的 MSI 中断

当 DMA 操作完成时, MSI 中断就会通知主机。主机在接收到该中断后, 轮询 DMA 读和写状态表以确定哪一个或多个条目已设置 done 位。该机制免于主机软件对状态表 done 位的持续轮询。使用此接口接收关于通过 TX Slave 接口生成根端口 MSI 或 MSI-X 中断的所需信息。

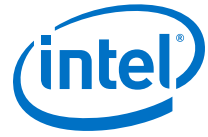


表 45. MSI 中断

信号	方向	说明
MSIIntfc_o[81:0]	输出	This bus provides the following MSI address, data, and enabled signals: <ul style="list-style-type: none"> MSIIntfc_o[81]: Master enable (主端口使能) MSIIntfc_o[80]: MSI enable (MSI 使能) MSIIntfc_o[79:64]: MSI data (MSI 数据) MSIIntfc_o[63:0]: MSI address (MSI 地址)
MSIXIntfc_o[15:0]	输出	提供 <i>PCI Local Bus Specification, Rev. 3.0</i> 中小节 6.8.2.3 <i>Message Control for MSI-X</i> 中定义的 MSI-X 的系统软件控制。域定义如下: <ul style="list-style-type: none"> MSIXIntfc_o[15]: 使能 MSIXIntfc_o[14]: Mask MSIXIntfc_o[13:11]: 保留 MSIXIntfc_o[10:0]: 列表大小
MSIControl_o[15:0]	输出	提供 <i>PCI Local Bus Specification, Rev. 3.0</i> 中小节 6.8.1.3 <i>Message Control for MSI</i> 所定义的 MSI 消息的系统控制。域定义如下: <ul style="list-style-type: none"> MSIControl_o[15:9]: 保留 MSIControl_o[8]: Per-Vector Masking 功能 MSIControl_o[7]: 64-Bit Address 功能 MSIControl_o[6:4]: 多消息使能 MSIControl_o[3:1]: MSI 消息功能 MSIControl_o[0]: MSI 使能
intx_req_i	输入	传统中断请求
intx_ack_o	输出	传统中断响应

5.5. 硬核 IP 重配置接口

硬核 IP 重配置接口是支持 10-bit 地址和 16-bit 数据总线的 Avalon-MM 从接口。您可使用该总线动态修改运行时为只读的配置寄存器值。在改变 Hard IP 只读配置寄存器的值以后，为确保系统正常运行，复位或重复 PCI Express 链路的器件枚举。

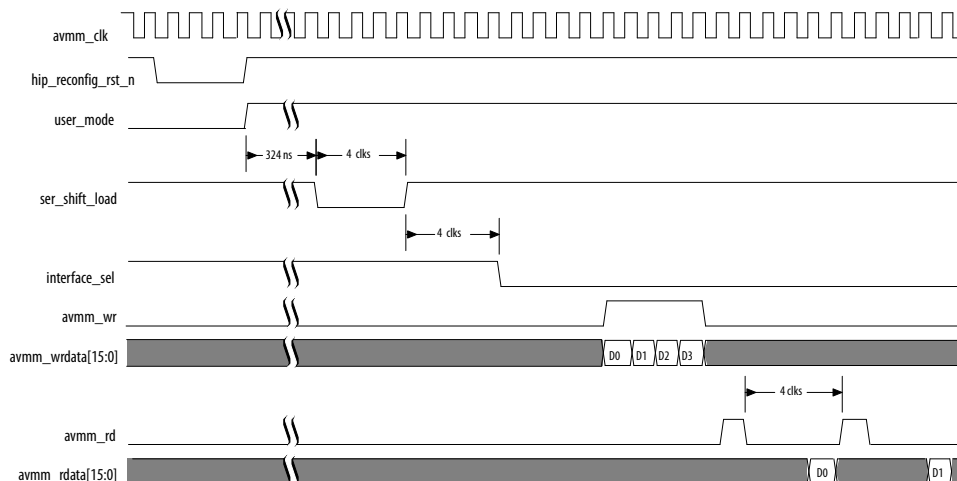
表 46. 硬核 IP 重配置信号

信号	方向	说明
hip_reconfig_clk	输入	重配置时钟。该时钟的频率范围是 100 – 125 MHz。
hip_reconfig_rst_n	输入	有效低电平 Avalon-MM 复位。根据 <i>Hard IP Reconfiguration Registers</i> 中描述的默认值，复位所有动态重配置寄存器。
hip_reconfig_address[9:0]	输入	10-bit 重配置地址。
hip_reconfig_read	输入	读信号。此接口未被流水线化。在开始另一个读操作前，必须等待 hip_reconfig_readdata[15:0] 从当前读取中返回。
hip_reconfig_readdata[15:0]	输出	16-bit 读数据。置位 hip_reconfig_read 后，hip_reconfig_readdata[15:0] 在第三个周期有效。
hip_reconfig_write	输入	写信号。
hip_reconfig_writedata[15:0]	输入	16-bit 写模型。

继续...

信号	方向	说明
hip_reconfig_byte_en[1:0]	输入	字节使能,当前未使用。
ser_shift_load	输入	更换到用户模式后,在第一次访问只读寄存器前必须立刻翻转该信号。切换到用户模式后,该信号应该保持置位至少 324ns。
interface_sel	输入	运行动态重配置时,必须置位选择器 (selector)。释放 ser_shift_load 后,驱动该信号低电平 4 个时钟周期。

图 28. 只读寄存器的硬核 IP 重配置总线时序



要了解关于 Avalon-MM 协议的详细信息,请参阅 *Avalon Interface Specifications* 中 *Avalon Memory Mapped Interfaces* 章节。

相关链接

[Avalon Interface Specifications](#)

关于 Avalon-MM 接口实现主,从组件读和写接口的信息。

5.6. 物理层接口信号

Intel 提供的是一个集合了事务,数据链路及物理层的方案。除了 Hard IP variation 文件, <variation>.v 或 .vhd 外, IP 参数编辑器生成一个 SERDES variation 文件, <variation>_serdes.v 或 .vhd。SERDES 实体包含于 PCI Express 的库文件中。

5.6.1. 串行数据信号

表 47. 1-Bit 接口信号

以下表格显示了用于 x8 IP 内核的信号。x1 IP 内核仅有 lane 0。x2 IP 内核仅有 lane 1 - 0。x4 IP 内核仅有 IP lane 3 - 0。

信号	方向	说明
tx_out[7:0]	输出	发送输出。这些信号为 lane 7 - 0 的串行输出。
rx_in[7:0]	输入	接收输入。这些信号是 lane 7 - 0 的串行输入。



请参阅 *Pin-out Files for Intel Devices* 获取 .pdf, .txt, 和 .xls 格式的 Intel 所有器件 pin-out 列表。

接收器 channel 6 个分为一组。对于 GX 器件, 左边最低的 6 个 channel 标记为 GXB_L0, 下一组为 GXB_L1, 以此类推。器件右边的 channel 被标记为 GXB_R0, GXB_R1, 以此类推。根据 *Pin-out Files for Intel Devices* 所指定, 务必将器件左边的 Hard IP for PCI Express 连接到器件左边正确的 channel。

相关链接

- Arria 10 器件中 Hard IP 的物理布局 (第 28 页)
图解带有 1-4 PCIe 硬核 IP 块的器件封装
- Pin-out Files for Intel Devices

5.6.2. PIPE 接口信号

这些 PIPE 信号可用于 Gen1, Gen2, 和 Gen3 variant, 因此既可使用串行接口也可使用 PIPE 接口仿真。使用 PIPE 接口的仿真速度较快, 因为 PIPE 仿真旁路 serdes 模型。默认情况下, PIPE 接口 8-bit 用于 Gen1 和 Gen2, 32-bit 用于 Gen3。即便您的设计中实际已有用于内部收发器的串行接口, 也可使用 PIPE 接口来仿真。然而, 硬件中不可使用 Hard IP PIPE 接口, 包括, 使用 SignalTap® II 嵌入式逻辑分析器探测这些信号。这些信号不是硬核 IP 的顶层信号。罗列在此以辅助调试链路训练问题。

注意: 根端口 BFM 旁路 Gen3 Phase 2 和 Phase 3 均衡。然而, 如果在第三方 BFM 指令下 Gen3 variant 可运行 Phase 2 和 Phase 3 均衡。

表 48. PIPE 接口信号

以下表格中, 包含 lane 编号 0 的信号也出现在 lane 1-7 中。这些信号仅用于仿真。Quartus Prime 软件编译中, 这些管道信号可以悬空。在 Qsys 中, 作为 PIPE 接口部分的信号其前缀为, hip_pipe。而包含在 PIPE 接口仿真中的信号其前缀为, hip_pipe_sim_pipe。

信号	方向	说明
currentcoeff0[17:0]	输出	对于 Gen3, 表示发送器将使用的系数。18 bits 指定以下系数: <ul style="list-style-type: none">[5:0]: C-1[11:6]: C0[17:12]: C+1
currentrxpreset0[2:0]	输出	对于 Gen3 的设计, 指定当前预设置。
eidleinfersel0[2:0]	输出	电气空闲进入推理机制选择。编码定义如下: <ul style="list-style-type: none">3'b0xx: 当前 LTSSM 状态中不需要电气空闲推理3'b100: 用于 Gen1 或 Gen2 的 128 us 窗口中 COM/SKP 有序集的缺失3'b101: 用于 Gen1 或 Gen2 的 1280 UI 间隔中 TS1/TS2 有序集的缺失3'b110: 用于 Gen1 的 2000 UI 间隔和用于 Gen2 的 16000 UI 间隔中电气空闲退出 (Electrical Idle Exit) 缺失3'b111: 用于 Gen1 的 128 us 窗口中电气空闲退出 (Electrical Idle Exit) 缺失
phystatus0	输入	PHY status <n> (PHY 状态)。该信号与几个 PHY 请求完成进行通信。
powerdown0[1:0]	输出	Power down <n> (断电)。该信号请求 PHY 将电源状态更改为指定状态 (P0, P0s, P1, 或 P2)。
rate[1:0]	输出	控制链路信号率。编码定义如下:
继续...		



信号	方向	说明
		<ul style="list-style-type: none"> 2'b00: Gen1 2'b01: Gen2 2'b10: Gen3 2'b11: Reserved
rxblkst0	输入	关于 Gen3 的操作，表示在接收方向中开启一个块。
rxdata0[31:0]	输入	接收数据。该总线接收 lane<n>上的数据。
rxdata0[3:0]	输入	用于接收数据符号的数据/控制位。Bit 0 对应 rxdata 最低电平顺序字节，诸如此类。值 0 表示一个数据字节。值 1 表示一个控制字节。仅 Gen1 和 Gen2。
rxelecidle0	输入	Receive electrical idle <n> (接收电气空闲)。置位后，表示电气空闲的检测。
rxpolarity0	输出	Receive polarity <n> (接收极性)。该信号指令 PHY 层倒转 8B/10B 接收器解码块极性。
rxstatus0[2:0]	输入	Receive status <n> (接收状态)。该信号为接收数据流和接收器检测编码接收状态和错误代码。
rxvalid0	输入	Receive valid <n> (接收有效)。该符号表示 rxdata<n>和 rxdata0 <n>上的符号块和有效数据。
sim_pipe_ltssmstate0[4:0]	输入和输出	<p>LTSSM 状态: LTSSM 状态机编码定义以下状态:</p> <ul style="list-style-type: none"> 5' b00000: Detect.Quiet (检测。静音) 5' b 00001: Detect.Active (检测。有效) 5' b00010: Polling.Active (轮询。有效) 5' b 00011: Polling.Compliance (轮询。合规) 5' b 00100: Polling.Configuration (轮询。配置) 5' b00101: Polling.Speed (轮询。速度) 5' b00110: Config.LinkwidthsStart (配置。链路宽度开启) 5' b 00111: Config.Linkaccept (配置。链路接受) 5' b 01000: Config.Lanenumaccept (配置。通道数接受) 5' b01001: Config.Lanenumwait (配置。通道数等待) 5' b01010: Config.Complete (配置。完成) 5' b 01011: Config.Idle (配置。空闲) 5' b01100: Recovery.Rcvlock (恢复。恢复时钟) 5' b01101: Recovery.Rcvconfig (恢复。恢复配置) 5' b01110: Recovery.Idle (恢复。空闲) 5' b 01111: L0 5' b10000: Disable (禁用) 5' b10001: Loopback.Entry (环回。进入) 5' b10010: Loopback.Active (环回。有效) 5' b10011: Loopback.Exit (环回。退出) 5' b10100: Hot.Reset (热。复位) 5' b10101: L0s 5' b11001: L2.transmit.Wake (L2.传输。唤醒) 5' b11010: Speed.Recovery (速度。恢复) 5' b11011: Recovery.Equalization, Phase 0 (恢复。均衡, Phase 0) 5' b11100: Recovery.Equalization, Phase 1 (恢复。均衡, Phase 1) 5' b11101: Recovery.Equalization, Phase 2 (恢复。均衡, Phase 2) 5' b11110: Recovery.Equalization, Phase 3 (恢复。均衡, Phase 3) 5' b11111: Recovery.Equalization, Done (恢复。均衡, 完成)
sim_pipe_pclk_in	输入	该时钟仅用于 PIPE 仿真，并产生自 refclk。它是用于 PIPE 模式仿真的 PIPE 接口时钟。
继续...		



信号	方向	说明
sim_pipe_rate[1:0]	输入	指定数据速率。2-bit 编码含义如下： <ul style="list-style-type: none"> 2' b00: Gen1 rate (2.5 Gbps) 2' b01: Gen2 rate (5.0 Gbps) 2' b1X: Gen3 rate (8.0 Gbps)
txblkst		用于 Gen3 操作，表示传输方向中开启一个模块。
txcompl0	输出	Transmit compliance <n> (发送兼容)。该信号强制编译模式中运行着的差异失效 (失效的 COM 字符)。
txdata0[31:0]	输出	发送数据。该总线发送 lane <n> 上的数据。
txdatak0[3:0]	输出	Transmit data control <n> (发送数据控制)。该信号用作 txdata <n> 的控制位。Bit 0 对应 rxdata 的最低电平顺序字节，诸如此类。值 0 表示一个数据字节。值 1 表示一个控制位。仅 Gen1 和 Gen2。
txdataskip0	输出	用于 Gen3 操作。允许 MAC 指令 TX 接口一个时钟周期内忽略 TX 数据接口。编码定义如下： <ul style="list-style-type: none"> 1' b0: TX 数据无效 1' b1: TX 数据有效
txdeemph0	输出	发送去加重选择。此信号的值是基于训练序列 (TS) 中从链路另一端接收到的指标。无需更改该值。
txdetectrx0	输出	Transmit detect receive <n> (发送检测接收)。此信号告知 PHY 层开启一个接收检测操作或开始环回。
txelecidle0	输出	Transmit electrical idle <n> (发送电气空闲)。此信号强制 TX 输出进入电气空闲。
txmargin0[2:0]	输出	发送 V _{OD} 合并选择。此信号的值基于来自于 Link Control 2 Register 的值。仅用于仿真。
txswing0	输出	置位后，表示发送器电压的全摆幅。置低时表示半幅。
txsynchd0[1:0]	输出	用于 Gen3 操作，指定块类型。编码定义如下： <ul style="list-style-type: none"> 2'b01: 有序集块 2'b10: 数据块

5.7. 测试信号

表 49. 测试接口信号

test_in 总线提供 IP 内核内部状态的运行时间控制和监控。

信号	方向	说明
test_in[31:0]	输入	test_in 总线位的定义如下。将该总线设置为 0x00000188。 <ul style="list-style-type: none"> [0]: 仿真模式。设置此信号到 1，以减少多个初始化计数器的值从而加速初始化。 [1]: 保留。必须设置为 1' b0。 [2]: 解扰模式能禁用。初始化期间此信号必须设置到 1，从而禁用数据扰频。可以在 Gen1 和 Gen2 端点和根端口仿真中使用该位，从而观察链路上已解扰数据。已解扰数据不可用于开放系统中，因为通常链路搭档会扰频数据。 [4:3]: 保留。必须设置为 2' b01。 [5]: 合规测试模式。将该位设置为 1'b0。将该位设置为 1'b1 以阻止 LTSSM 进入合规模式。翻转此位能控制进入或退出合规状态，使能 Gen1, Gen2 和 Gen3 合规码型的传输。 [6]: 强制进入合规模式，当 polling.active (轮询。有效) 状态中达到超时，且非全部 lane 都检测到退出条件。

继续...



信号	方向	说明
		<ul style="list-style-type: none"> [7]: 禁用低功耗状态协商。Intel 建议设置此位。 [8]: 将该位设置为 1'b1。 [31:9]: 保留。设置为全 0。
currentspeed[1:0]	输出	显示 PCIe 链路的当前速度。编码定义如下: <ul style="list-style-type: none"> 2b' 00: 未定义 2b' 01: Gen1 2b' 10: Gen2 2b' 11: Gen3

相关链接

PIPE 接口信号 (第 53 页)

5.8. Arria 10 开发套件导管接口

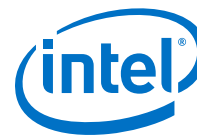
Arria 10 开发套件导管接口信号是可选信号，它允许将您的设计连接到 Arria 10 FPGA 开发套件。通过在 GUI 组件的 **Configuration, Debug, and Extension Options** 选项卡中选择 **Enable Arria 10 FPGA Development Kit connection** 使能该接口。devkit_status 输出端口包含有助于调试的信号。

表 50.

信号名称	方向	说明
devkit_status[255:0]	输出	devkit_status[255:0] 总线包括以下状态信号: <ul style="list-style-type: none"> devkit_status[1:0]: current_speed devkit_status[2]: derr_cor_ext_rcv devkit_status[3]: derr_cor_ext_rpl devkit_status[4]: derr_err devkit_status[5]: rx_par_err devkit_status[7:6]: tx_par_err devkit_status[8]: cfg_par_err devkit_status[9]: dlup devkit_status[10]: dlup_exit devkit_status[11]: evl28ns devkit_status[12]: evlus devkit_status[13]: hotrst_exit devkit_status[17:14]: int_status[3:0] devkit_status[18]: l2_exit devkit_status[22:19]: lane_act[3:0] devkit_status[27:23]: ltssmstate[4:0] devkit_status[35:28]: ko_cpl_spc_header[7:0] devkit_status[47:36]: ko_cpl_spc_data[11:0] devkit_status[48]: rxfc_cplbuf_ovf devkit_status[49]: reset_status devkit_status[255:50]: Reserved
devkit_ctrl[255:0]	输入	devkit_ctrl[255:0] 总线包括以下状态信号。您可以选择性地将这些管脚连接到片上切换开关来进行 PCI-SIG 合规测试，例如旁路合规测试。
继续...		



信号名称	方向	说明
		<ul style="list-style-type: none">devkit_ctrl[0]:test_in[0] is typically set to 1'b0devkit_ctrl[4:1]:test_in[4:1] is typically set to 4'b0100devkit_ctrl[6:5]:test_in[6:5] is typically set to 2'b01devkit_ctrl[31:7]:test_in[31:7] is typically set to 25'h3devkit_ctrl[63:32]:is typically set to 32'b0devkit_ctrl[255:64]:is typically set to 192'b0



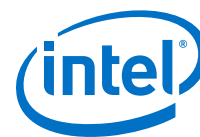
6. 寄存器

6.1. 配置空间寄存器与 PCIe 规范的一致性

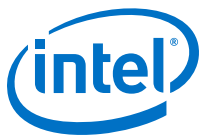
表 51. 配置空间性能结构与 PCIe 基本规范说明的一致性

对于 Type 0 和 Type 1 配置空间头 (Configuration Space Header)，当值相异时，每个入口的第一行罗列 Type 0 的值，第二行罗列 Type 1 的值。

字节地址	硬核 IP 配置空间寄存器	PCIe 规范中的对应部分
0x000:0x03C	PCI Header Type 0 Configuration Registers	Type 0 Configuration Space Header
0x000:0x03C	PCI Header Type 1 Configuration Registers	Type 1 Configuration Space Header Type 1 配置空间不用于带有 DMA 的 Avalon-MM 接口
0x040:0x04C	保留	N/A
0x050:0x05C	MSI Capability Structure	MSI Capability Structure
0x068:0x070	MSI-X Capability Structure	MSI-X Capability Structure
0x070:0x074	保留	N/A
0x078:0x07C	Power Management Capability Structure	PCI Power Management Capability Structure
0x080:0x0B8	PCI Express Capability Structure	PCI Express Capability Structure
0x0B8:0x0FC	保留	N/A
0x094:0x0FF	Root Port	N/A
0x100:0x16C	Virtual Channel Capability Structure (保留)	Virtual Channel Capability
0x170:0x17C	保留	N/A
0x180:0x1FC	Virtual channel arbitration table (保留)	VC Arbitration Table (VC 仲裁表)
0x200:0x23C	Port VC0 arbitration table (保留)	Port Arbitration Table (端口仲裁表)
0x240:0x27C	Port VC1 arbitration table (保留)	Port Arbitration Table
0x280:0x2BC	Port VC2 arbitration table (Reserved)	Port Arbitration Table
0x2C0:0x2FC	Port VC3 arbitration table (保留)	Port Arbitration Table
0x300:0x33C	Port VC4 arbitration table (Reserved)	Port Arbitration Table
0x340:0x37C	Port VC5 arbitration table (保留)	Port Arbitration Table
0x380:0x3BC	Port VC6 arbitration table (保留)	Port Arbitration Table
0x3C0:0x3FC	Port VC7 arbitration table (保留)	Port Arbitration Table
0x400:0x7FC	保留	PCIe 规范对应部分的名称
0x800:0x834	Advanced Error Reporting AER (可选)	Advanced Error Reporting Capability (高级错误报告能力)
继续...		

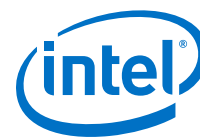


字节地址	硬核 IP 配置空间寄存器	PCIe 规范中的对应部分
0x838:0xFFF	保留	N/A
配置空间寄存器域总览		
0x000	Device ID, Vendor ID	Type 0 Configuration Space Header Type 1 Configuration Space Header Type 1 不用于带有 DMA 的 Avalon-MM 接口
0x004	Status, Command	Type 0 Configuration Space Header Type 1 Configuration Space Header Type 1 不用于带有 DMA 的 Avalon-MM 接口
0x008	Class Code, Revision ID	Type 0 Configuration Space Header Type 1 Configuration Space Header Type 1 不用于带有 DMA 的 Avalon-MM 接口
0x00C	BIST, Header Type, Primary Latency Timer, Cache Line Size	Type 0 Configuration Space Header Type 1 Configuration Space Header Type 1 不用于带有 DMA 的 Avalon-MM 接口
0x010	Base Address 0	Base Address Registers
0x014	Base Address 1	Base Address Registers
0x018	Base Address 2 Secondary Latency Timer, Subordinate Bus Number, Secondary Bus Number, Primary Bus Number	Base Address Registers Secondary Latency Timer, Type 1 Configuration Space Header, Primary Bus Number
0x01C	Base Address 3 Secondary Status, I/O Limit, I/O Base	Base Address Registers Secondary Status Register, Type 1 Configuration Space Header
0x020	Base Address 4 Memory Limit, Memory Base	Base Address Registers Type 1 Configuration Space Header
0x024	Base Address 5 Prefetchable Memory Limit, Prefetchable Memory Base	Base Address Registers Prefetchable Memory Limit, Prefetchable Memory Base
0x028	保留 Prefetchable Base Upper 32 Bits	N/A Type 1 Configuration Space Header
0x02C	Subsystem ID, Subsystem Vendor ID Prefetchable Limit Upper 32 Bits	Type 0 Configuration Space Header Type 1 Configuration Space Header
0x030	I/O Limit Upper 16 Bits, I/O Base Upper 16 Bits	Type 0 Configuration Space Header Type 1 Configuration Space Header
0x034	Reserved, Capabilities PTR	Type 0 Configuration Space Header Type 1 Configuration Space Header
0x038	保留	N/A
0x03C	Interrupt Pin, Interrupt Line Bridge Control, Interrupt Pin, Interrupt Line	Type 0 Configuration Space Header Type 1 Configuration Space Header
0x050	MSI-Message Control Next Cap Ptr Capability ID	MSI and MSI-X Capability Structures
0x054	Message Address	MSI and MSI-X Capability Structures
0x058	Message Upper Address	MSI and MSI-X Capability Structures
0x05C	Reserved Message Data	MSI and MSI-X Capability Structures
继续...		



字节地址	硬核 IP 配置空间寄存器	PCIe 规范中的对应部分
0x068	MSI-X Message Control Next Cap Ptr Capability ID	MSI and MSI-X Capability Structures
0x06C	MSI-X Table Offset BIR	MSI and MSI-X Capability Structures
0x070	Pending Bit Array (PBA) Offset BIR	MSI and MSI-X Capability Structures
0x078	Capabilities Register Next Cap PTR Cap ID	PCI Power Management Capability Structure
0x07C	Data PM Control/Status Bridge Extensions Power Management Status & Control	PCI Power Management Capability Structure
0x800	PCI Express Enhanced Capability Header	Advanced Error Reporting Enhanced Capability Header
0x804	Uncorrectable Error Status Register	Uncorrectable Error Status Register
0x808	Uncorrectable Error Mask Register	Uncorrectable Error Mask Register
0x80C	Uncorrectable Error Severity Register	Uncorrectable Error Severity Register
0x810	Correctable Error Status Register	Correctable Error Status Register
0x814	Correctable Error Mask Register	Correctable Error Mask Register
0x818	Advanced Error Capabilities and Control Register	Advanced Error Capabilities and Control Register
0x81C	Header Log Register	Header Log Register
0x82C	Root Error Command	Root Error Command Register
0x830	Root Error Status	Root Error Status Register
0x834	Error Source Identification Register Correctable Error Source ID Register	Error Source Identification Register

[相关链接](#)[PCI Express Base Specification 3.0](#)



6.2. Type 0 配置空间寄存器

图 29. Type 0 配置空间寄存器 - 字节地址偏移和布局

端点储存配置数据在 Type 0 配置空间中。配置空间寄存器与 PCIe 规范的一致性 (第 58 页) 罗列了 *PCI Express Base Specification* 中介绍这些寄存器的相应部分。

	31	24	23	16	15	8	7	0	
0x000	Device ID					Vendor ID			
0x004	Status					Command			
0x008	Class Code						Revision ID		
0x00C	0x00		Header Type			0x00		Cache Line Size	
0x010	BAR Registers								
0x014	BAR Registers								
0x018	BAR Registers								
0x01C	BAR Registers								
0x020	BAR Registers								
0x024	BAR Registers								
0x028	Reserved								
0x02C	Subsystem Device ID					Subsystem Vendor ID			
0x030	Expansion ROM Base Address								
0x034	Reserved						Capabilities Pointer		
0x038	Reserved								
0x03C	0x00				Interrupt Pin		Interrupt Line		

6.3. Type 1 配置空间寄存器

图 30. Type 1 配置空间寄存器（根端口）

31	24 23		16 15		8 7		0
0x0000	Device ID			Vendor ID			
0x0004	Status			Command			
0x0008	Class Code				Revision ID		
0x000C	BIST	Header Type		Primary Latency Timer		Cache Line Size	
0x0010	BAR Registers						
0x0014	BAR Registers						
0x0018	Secondary Latency Timer		Subordinate Bus Number		Secondary Bus Number		Primary Bus Number
0x001C	Secondary Status			I/O Limit		I/O Base	
0x0020	Memory Limit			Memory Base			
0x0024	Prefetchable Memory Limit			Prefetchable Memory Base			
0x0028	Prefetchable Base Upper 32 Bits						
0x002C	Prefetchable Limit Upper 32 Bits						
0x0030	I/O Limit Upper 16 Bits			I/O Base Upper 16 Bits			
0x0034	Reserved					Capabilities Pointer	
0x0038	Expansion ROM Base Address						
0x003C	Bridge Control			Interrupt Pin		Interrupt Line	

6.4. PCI Express 性能结构

以下为最基本性能结构布局。请参阅 *PCI Express Base Specification* 了解更多关于这些寄存器的信息。

图 31. MSI 性能结构

	31	24 23	16 15	8 7	0
0x050	Message Control Configuration MSI Control Status Register Field Descriptions			Next Cap Ptr	Capability ID
0x054	Message Address				
0x058	Message Upper Address				
0x05C	Reserved			Message Data	

图 32. MSI-X 性能结构

	31	24 23	16 15	8 7	3 2	0
0x068	Message Control					
	Next Cap Ptr				Capability ID	
0x06C	MSI-X Table Offset					MSI-X Table BAR Indicator
0x070	MSI-X Pending Bit Array (PBA) Offset					MSI-X Pending Bit Array - BAR Indicator

图 33. 电源管理性能结构 - 字节地址偏移和布局

	31	24	23	16	15	8	7	0	
0x078	Capabilities Register					Next Cap Ptr		Capability ID	
0x07C	Data		PM Control/Status Bridge Extensions		Power Management Status and Control				

图 34. PCI Express AER 扩展性能结构

Byte Offset	31:24	23:16	15:8	7:0
0x800	PCI Express Enhanced Capability Register			
0x804	Uncorrectable Error Status Register			
0x808	Uncorrectable Error Mask Register			
0x80C	Uncorrectable Error Severity Register			
0x810	Correctable Error Status Register			
0x814	Correctable Error Mask Register			
0x818	Advanced Error Capabilities and Control Register			
0x81C	Header Log Register			
0x82C	Root Error Command Register			
0x830	Root Error Status Register			
0x834	Error Source Identification Register		Correctable Error Source Identification Register	

图 35. PCI Express 性能结构 - 字节地址偏移和布局

以下表格显示了 PCI Express 性能结构，寄存器不可用于已保留的器件。

注意: 带有 DMA 的 Avalon-MM 接口不支持根端口

	31	24 23	16 15	8 7	0
0x080	PCI Express Capabilities Register			Next Cap Pointer	PCI Express Capabilities ID
0x084	Device Capabilities				
0x088	Device Status			Device Control	
0x08C	Link Capabilities				
0x090	Link Status			Link Control	
0x094	Slot Capabilities				
0x098	Slot Status			Slot Control	
0x09C	Root Capabilities			Root Control	
0x0A0	Root Status				
0x0A4	Device Compatibilities 2				
0x0A8	Device Status 2			Device Control 2	
0x0AC	Link Capabilities 2				
0x0B0	Link Status 2			Link Control 2	
0x0B4	Slot Capabilities 2				
0x0B8	Slot Status 2			Slot Control 2	

相关链接

- [PCI Express Base Specification 3.0](#)
- [PCI Local Bus Specification](#)

6.5. Intel 定义的 VSEC 寄存器

图 36. VSEC 寄存器

此扩展的性能结构支持 Configuration via Protocol (CvP)编程和详细的内部错误报告。

	31	20 19	16 15	8 7	0
0x200	Next Capability Offset		Version	Intel-Defined VSEC Capability Header	
0x204	VSEC Length		VSEC Revision	VSEC ID Intel-Defined, Vendor-Specific Header	
0x208	Intel Marker				
0x20C	JTAG Silicon ID DW0 JTAG Silicon ID				
0x210	JTAG Silicon ID DW1 JTAG Silicon ID				
0x214	JTAG Silicon ID DW2 JTAG Silicon ID				
0x218	JTAG Silicon ID DW3 JTAG Silicon ID				
0x21C	CvP Status			User Device or Board Type ID	
0x220	CvP Mode Control				
0x224	CvP Data2 Register				
0x228	CvP Data Register				
0x22C	CvP Programming Control Register				
0x230	Reserved				
0x234	Uncorrectable Internal Error Status Register				
0x238	Uncorrectable Internal Error Mask Register				
0x23C	Correctable Internal Error Status Register				
0x240	Correctable Internal Error Mask Register				

表 52. Intel 定义的 VSEC 性能寄存器, 0x200

Intel 定义供应商的指定扩展性能。此扩展性能结构支持 Configuration via Protocol (CvP)编程以及详细的内部错误报告。

Bits	寄存器说明	值	访问权限
[15:0]	PCI Express 扩展性能 ID。Intel 定义的 VSEC 性能 ID 值。	0x000B	RO (只读)
[19:16]	版本。Intel 指定的 VSEC 版本值。	0x1	RO
[31:20]	下一个性能偏移。下一性能结构起始地址实现, 如有。	变量	RO

表 53. Intel 定义供应商指定头

例化硬核 IP 时, 可指定这些值。这些寄存器在运行时为只读。

Bits	寄存器说明	值	访问权限
[15:0]	VSEC ID。用户可配置 VSEC ID。	用户输入	RO
[19:16]	VSEC Revision。用户可配置 VSEC 版本。	变量	RO
[31:20]	VSEC Length。该结构以字节为单位的总长度。	0x044	RO



表 54. IntelMarker 寄存器

Bits	寄存器说明	值	访问权限
[31:0]	Intel Marker。该只读寄存器是一个额外 marker。如果使用标准的 Intel 编程软件配置具有 CvP 的器件，该 marker 提供一个编程软件读取的值以确保通过正确的 VSEC 进行操作。	器件值	RO

表 55. JTAG Silicon ID 寄存器

Bits	寄存器说明	值	访问权限
[127:96]	JTAG Silicon ID DW3	特定应用	RO
[95:64]	JTAG Silicon ID DW2	特定应用	RO
[63:32]	JTAG Silicon ID DW1	特定应用	RO
[31:0]	JTAG Silicon ID DW0。这是 JTAG Silicon ID，CvP 编程软件读取它以决定使用的是正确的 SRAM 目标文件(.sof)。	特定应用	RO

表 56. 用户器件或电路板类型 ID 寄存器

Bits	寄存器说明	值	访问权限
[15:0]	可配置器件或电路板类型 ID 为 CvP 指定正确的.sof。	变量	RO

6.6. CvP 寄存器

表 57. CvP 寄存器

CvP Status 寄存器允许软件监控 CvP 状态信号。

Bits	寄存器说明	复位值	访问权限
[31:26]	保留	0x00	RO (只读)
[25]	PLD_CORE_READY。来自于 FPGA 架构。用于调试的状态位。	变量	RO
[24]	PLD_CLK_IN_USE。从时钟切换模块到架构。用于调试的状态位。	变量	RO
[23]	CVP_CONFIG_DONE。表示 FPGA 控制块已完成通过 CvP 的器件配置且无错误。	变量	RO
[22]	保留	变量	RO
[21]	USERMODE。表示可配置 FPGA 架构是否处于用户模式。	变量	RO
[20]	CVP_EN。表示 FPGA 控制块是否已使能 CvP 模式。	变量	RO
[19]	CVP_CONFIG_ERROR。反映来自 FPGA 控制块信号的值，并通过软件查看以确定配置期间是否有错误。	变量	RO
[18]	CVP_CONFIG_READY。表示来自 FPGA 控制块信号的值，软件在编程算法期间查看该值。	变量	RO
[17:0]	保留	变量	RO

表 58. CvP 模式控制

CvP Mode Control 寄存器提供 CvP 操作的全局控制。

Bits	寄存器说明	复位值	访问权限
[31:16]	保留	0x0000	RO (只读)
[15:8]	CVP_NUMCLKS。 为每个 CvP 数据写发送的时钟数目。根据您的配置映像，将此域设置为以下其中一个值： <ul style="list-style-type: none"> 0x01: 用于未压缩也未加密的映像 0x04: 用于未压缩但已加密的映像 0x08: 用于所有已压缩映像 	0x00	RW (读, 写)
[7:3]	保留	0x0	RO
[2]	CVP_FULLCONFIG。需要 FPGA 控制块重配置整个 FPGA, 包括 Arria 10 Hard IP for PCI Express, 使 PCIe 链路下行。	1' b0	RW
[1]	HIP_CLK_SEL。USER_MODE = 1 和 PLD_CORE_READY = 1 时, 在 PMA 与架构时钟间选择。编码定义如下: <ul style="list-style-type: none"> 1: 选择来自 PMA 的内部时钟, 需用于 CVP_MODE。 0: 选择来自软逻辑架构的时钟。只有当通过连接到正确时钟的配置文件在 USER_MODE 中配置架构时, 才能使用此设置。 确保 CvP 期间没有时钟切换, 只有当 Hard IP for PCI Express 空闲时间已达 10 μ s 时才能更改该值, 并且需要在更改后等待 10 μ s 再恢复活动。	1' b0	RW
[0]	CVP_MODE。控制 IP 内核是否处于 CVP_MODE 或普通模式。编码定义如下: <ul style="list-style-type: none"> 1: CVP_MODE 有效。到 FPGA 控制块的信号有效并且全部 TLP 被路由到配置空间。如果 CVP_EN = 0, 不可使能 CVP_MODE。 0: IP 内核处于普通模式并且 TLP 被路由到 FPGA 架构。 	1' b0	RW

表 59. CvP 数据寄存器

以下表格定义了 CvP Data 寄存器。对于 64-bit 数据, 可选的 CvP Data2 储存数据的上层 32 bits。编程软件写配置数据到这些寄存器中。如果每次写这些寄存器都设置 FPGA 控制块的数据输出, 就会按照 CvP Mode Control 寄存器中 CVP_NUM_CLKS 域所指定的生成 $<n>$ 个时钟周期到 FPGA 控制块中。软件必须确保存储器写 dword 中所有字节都已使能。可通过配置写访问该寄存器, 或者, 当处于 CvP 模式时, 也可以使用存储器写将这些寄存器写入由器件存储器空间 BAR 所定义的任何地址。使用存储器写将允许比配置写更高的吞吐量。

Bits	寄存器说明	复位值	访问权限
[31:0]	配置数据的上 32 位被传输到 FPGA 控制块以用于配置器件。您可以选择 32 位或 64 位数据。	0x00000000	RW
[31:0]	配置数据的下 32 位被传输到 FPGA 控制块以配置器件。	0x00000000	RW

表 60. CvP 编程控制寄存器

此寄存器由编程软件写入以控制 CvP 编程。

Bits	寄存器说明	复位值	访问权限
[31:2]	保留	0x0000	RO
[1]	START_XFER。设置 CvP 输出到 FPGA 控制块表示传输开始。	1' b0	RW
[0]	CVP_CONFIG。置位后, 指令 FPGA 控制块开始通过 CvP 传输。	1' b0	RW

6.7. 不可纠正的内部错误掩码寄存器

表 61. 不可纠正的内部错误掩码寄存器

Uncorrectable Internal Error Mask 寄存器控制内部不可更正错误的转发。除了 CvP 模式中检测到的配置错误，所有错误都很严重，且会导致器件或 PCIe 链路状态不一致。取决于编程软件的设计，CvP 模式中检测到的配置错误或许可以更正。访问代码 RWS 代表 Read Write Sticky（读写粘附），表示在 IP 内核软复位后该值被保留。

位	寄存器说明	复位值	通路
[31:12]	已保留。	1b' 0	RO（只读）
[11]	已发布的 RX 缓冲器掩码和完成上溢错误。	1b' 1	RWS（读写粘附）
[10]	已保留	1b' 0	RO
[9]	配置空间到 TX 总线接口上检测到的奇偶校验错误掩码。	1b' 1	RWS
[8]	TX 到配置空间总线接口上检测到的奇偶校验错误掩码	1b' 1	RWS
[7]	在 TX 传输层检测到的奇偶校验错误掩码。	1b' 1	RWS
[6]	已保留	1b' 0	RO
[5]	CvP 模式中检测到的配置错误的掩码。	1b' 0	RWS
[4]	TX 数据链路生成期间检测到的数据奇偶校验错误掩码。	1b' 1	RWS
[3]	RX 到配置空间总线接口上检测到的数据奇偶校验错误掩码	1b' 1	RWS
[2]	输入到 RX 缓冲上检测到的数据奇偶校验错误掩码。	1b' 1	RWS
[1]	重试缓冲器不可纠正 ECC 错误掩码。	1b' 1	RWS
[0]	RX 缓冲器不可纠正 ECC 错误掩码。	1b' 1	RWS

6.8. 不可纠正的内部错误状态寄存器

表 62. 不可纠正的内部错误状态寄存器

此寄存器报告内部检测到的错位为不可纠正。当具体错误被 Uncorrectable Internal Error Mask 寄存器使能时，这些错误将作为 *PCI Express Base Specification 3.0* 中所定义的不可纠正错误被处理。此寄存器仅用于调试。它只能用于观察行为，而不能用来驱动自定义逻辑。访问权限代码 RW1CS 代表 Read Write 1 到 Clear Sticky（清除黏性）。

位	寄存器说明	复位值	通路
[31:12]	已保留。	0	RO（只读）
[11]	设置后，表示处于已发布请求或完成中的 RX 缓冲上溢条件	0	RW1CS
[10]	已保留。	0	RO
[9]	设置后，表示在配置空间到 TX 总线接口上检测到奇偶错误	0	RW1CS
[8]	设置后，表示 TX 到配置空间总线接口上检测到奇偶校验错误	0	RW1CS
[7]	设置后，表示在 TX TLP 中检测到奇偶校验错误且该 TLP 没有被发送。	0	RW1CS
[6]	设置后，表示应用层已检测到一个不可纠正的内部错误。	0	RW1CS
[5]	设置后，表示 CvP 模式中检测到配置错误，且报告为不可纠正。每当 CVP_CONFIG_ERROR 出现在 CVP_MODE 中，就设置为此位。	0	RW1CS
[4]	设置后，表示 TX 数据链路层检测到奇偶校验错误。	0	RW1CS
继续...			



位	寄存器说明	复位值	通路
[3]	设置后，表示 RX 到配置空间总线接口上检测到奇偶校验错误。	0	RW1CS
[2]	设置后，表示输入到 RX 缓冲上检测到奇偶校验错误。	0	RW1CS
[1]	设置后，表示重试缓冲器不可纠正 ECC 错误。	0	RW1CS
[0]	设置后，表示 RX 缓冲器不可纠正 ECC 错误。	0	RW1CS

[相关链接](#)

[PCI Express Base Specification 3.0](#)

6.9. 可纠正的内部错误掩码寄存器

表 63. 可纠正的内部错误掩码寄存器

Correctable Internal Error Mask 寄存器控制内部可纠正错误的转发。此寄存器仅用于调试。

Bits	寄存器说明	复位值	访问权限
[31:7]	已保留。	0	RO (只读)
[6]	应用层报告已纠正内部错误的掩码。	1	RWS (读写粘附)
[5]	CvP 模式中检测到的配置错误掩码。	0	RWS
[4:2]	已保留。	0	RO
[1]	重试缓冲器可纠正 ECC 错误掩码。	1	RWS
[0]	RX 缓冲器可纠正 ECC 错误掩码。	1	RWS

6.10. 可纠正的内部错误状态寄存器

表 64. 可纠正的内部错误状态寄存器

Correctable Internal Error Status 寄存器报告内部检测到的错误为可纠正。当这些具体错误被 Correctable Internal Error Mask 寄存器使能时，就将作为 *PCI Express Base Specification 3.0* 所定义的可纠正内部错误被转发。此寄存器仅用于调试。此寄存器仅用来观察行为，而不能用来驱动自定义逻辑。

位	寄存器说明	复位值	通路
[31:7]	已保留。	0	RO (只读)
[6]	由应用层报告的已纠正内部错误。	0	RW1CS
[5]	设置后，表示 CvP 模式中检测到配置错误，且被报告为可纠正。每当 CVP_CONFIG_ERROR 出现在 CVP_MODE 中，就设置为此位。	0	RW1CS
[4:2]	已保留。	0	RO (只读)
[1]	设置后，重试缓冲可纠正 ECC 错误状态显示有一个错误。	0	RW1CS
[0]	设置后，RX 缓冲可纠正 ECC 错误状态显示有一个错误。	0	RW1CS

[相关链接](#)

[PCI Express Base Specification 3.0](#)

6.11. DMA 描述符控制器寄存器

DMA 描述符控制器管理读和写 DMA 操作。描述符控制器支持读和写 DMA 的描述符多达 128 个。在嵌入式 CPU 上运行的主机软件根据 PCI Express 主存储器中描述符列表的位置和大小编程描述符内部寄存器。。DMA 描述符控制器指令 Read DMA 将该列表复制到其内部的 FIFO。当 DMA 描述符控制器被例化为单独的组件，就会驱动 RdDmaRxData_i[159:0]和 WrDmaRxData_i[159:0]总线上的列表条目。当 DMA 描述符控制器被嵌入到 Avalon-MM DMA 桥，就会驱动内部管道接口（internal conduit interface）上的该信息。

Read DMA 将数据从 PCIe 地址空间传输到 Avalon-MM 地址空间。它在 PCIe 链路上发布存储器读 TLP。并将返回数据写入 Avalon-MM 地址空间的存储器中。源地址是 PCIe 地址空间数据的地址。目的地址在 Avalon-MM 地址空间中。

Write DMA 从 Avalon-MM 地址空间读取数据，并写入 PCIe 地址空间。它在 PCIe 链路上发布存储器写 TLP。源地址在 Avalon-MM 地址空间。目的地址在 PCIe 地址空间。

DMA 描述符控制器把读和写描述符的完成状态记录在各自的状态列表中。每个列表具有 128 个 dword 条目与 128 个描述符相对应，描述符控制器写一个 1 到状态 dword 的 done 位以表示成功完成。描述符控制器还为最后的描述符发送 MSI 中断。接收此 MSI 后，主机软件可轮询 done 位以确定状态。在存储器中状态列表优先于描述符列表。因为每个描述符完成时，描述符控制器不会写 done 位或者发送 MSI。它仅仅为 ID 储存于 RD_DMA_LASTPTR 或 WR_DMA_LAST_PTR 寄存器的描述符写 done 位或者发送 MSI。描述符控制器支持无序完成。因此，在所有描述完成之前可能被设为 done 位。

128 个描述符的状态条目存储在 128 个连续的 dword 中，这 128 个连续的 dword 由已编程到 RC Read Descriptor Base 和 RC Write Descriptor Base 寄存器的值指定。这些值中位于偏移 0x200 的状态条目编程到 RC Read Descriptor Base 和 RC Write Descriptor Base 寄存器后，就立刻储存当前的描述符。状态和描述符列表必须位于根复合存储器的 32-byte 边界。

注意:

例如，如果指定 128 个描述符并全部执行，则写 127 个到 RD_DMA_LAST_PTR 或 WR_DMA_LAST_PTR 寄存器以开启 DMA。当描述符 127 完成时，DMA 描述符控制器才写 done 位。要得到中间状态更新，主机软件应该把多个 ID 写入最后一个指针寄存器。例如，要在 128 个读描述符完成一半时得到中间状态更新，主机软件应该完成以下序列：

1. 编程 RD_DMA_LAST_PTR = 63。
2. 编程 RD_DMA_LAST_PTR = 127。
3. 轮询用于读描述符 63 的状态 dword。
4. 轮询用于读描述符 127 的状态 dword。

在支持无序完成的系统中，描述符控制器会无序地完成描述符。因此，已储存描述符 <n> 的 done 状态并不一定表示描述符 <n-1> 和 <n-2> 也已经完成。您必须通过把每个描述符的描述符 ID 写入 RD_DMA_LAST_PTR 或 WR_DMA_LAST_PTR 以请求每个描述符的完成状态。而许多商用系统根端口基于主机存储器通道的已优化访问权限返回无序读完成。



6.11.1. 读 DMA 描述符控制器寄存器

以下表格介绍内部 DMA 描述符控制器中的寄存器。当外部例化 DMA 描述符控制器后，就可通过 BAR 访问这些寄存器。一定要把偏移添加到读控制器基地址中。当内部例化描述符控制器后，就通过 BAR0 访问这些寄存器。读控制器在偏移 0x0000。

地址偏移	寄存器	访问权限	说明
0x0000	RC Read Status and Descriptor Base (Low)	R/W	指定根复合存储器中读状态基址的下 32 位及描述符列表。此地址必须在 32 字节边界。在偏移 0x4 编程上 32 位后，内部软件必须编程此寄存器。更改 RC Read Status and Descriptor Base (Low) 基地址，就必须用完 RD_TABLE_SIZE 指定的所有描述符。
0x0004	RC Read Status and Descriptor Base (High)	R/W	指定根复合存储器中读状态基址上 32 位及描述符列表。在编程此寄存器下 32 位以前，软件必须编程此寄存器。
0x0008	EP Read Descriptor FIFO Base (Low)	RW	指定端点存储器中读描述符 FIFO 基址的下 32 bits。Read DMA 从根复合存储器取回描述符。此地址一定是通过 Read DMA Avalon-MM 主端口看到的描述符控制器 Read 描述符列表中 Avalon-MM 从端口的 Avalon-MM 地址。编程偏移 0x8 上 32 bits 后就必须编程此寄存器。
0x000C	EP Read Descriptor FIFO Base (High)	RW	指定端点 Avalon-MM 存储器读描述符列表中指定基址的上 32 bits。Read DMA 从根复合存储器取回描述符，并把描述符写入此处的 FIFO。此地址必定是通过 Read DMA Avalon-MM 主端口看到的描述符控制器 Read 描述符列表中 Avalon-MM 从端口的 Avalon-MM 地址。必须先编程此寄存器再编程此寄存器的下 32 位。
0x0010	RD_DMA_LAST_PTR	RW	<p>读取时，返还最后一个描述符请求的 ID。如果无未处理的 DMA 请求或者 DMA 处于复位中，就返回一个值 0xFF。</p> <p>被写入时，指定所请求最后一个描述符的 ID。读取值和写入值的不同在于待处理的描述符个数。</p> <p>例如，读取值为 4，所请求的最后一个描述符就为 4。要多指定 5 个描述符，软件应该把 9 写入 RD_DMA_LAST_PTR 寄存器寄存器。DMA 就多执行 5 个描述符。</p> <p>要读 DMA 记录每个描述符的 done 位，编程此寄存器每次仅传输一个描述符。</p> <p>描述符 ID 到达 RD_TABLE_SIZE 后，环回到 0。例如，如果 RD_TABLE_SIZE 值读为 126，而您想再多执行 3 个描述符，软件必须写 127，然后把 1 写入 RD_DMA_LAST_PTR 寄存器。</p>
0x0014	RD_TABLE_SIZE	RW	指定 Read 描述符列表大小。设置描述符个数-1。默认情况下，RD_TABLE_SIZE 设置为 127。该值指定最后一个 Descriptor ID。要更改 RC Read Status and Descriptor Base (Low) 基地址，必须完 RD_TABLE_SIZE 指定的所有描述符。
0x0018	RD_CONTROL	RW	<p>[31:1]保留。</p> <p>[0]Done。设置后，描述符控制器为状态列表里的每个描述符写 Done 位。在完成最后一个描述符后，描述符控制器发送单个 MSI 中断。如果最后一个描述符没有生成 Done，就由 RD_DMA_LAST_PTR 指定。</p>

6.11.2. 写 DMA 描述符控制器寄存器

以下表格介绍内部 DMA 描述符控制器中的寄存器。当外部例化 DMA 描述符控制器后，就可通过 BAR 访问这些寄存器。一定要把偏移添加到读控制器基地址中。当内部例化描述符控制器后，就通过 BAR0 访问这些寄存器。读控制器在偏移 0x0100。

地址偏移	寄存器	访问权限	说明
0x0100	RC Write Status and Descriptor Base (Low)	R/W	指定根复合存储器中读状态基址下 32-bit 及描述符列表。此地址必须位于 32 字节边界。编程位于偏移 0x104 的上 32 bits 后，内部软件必须编程此寄存器。要更改 RC Write Status and Descriptor Base (Low) 基地址，就必须用完 RD_TABLE_SIZE 指定的所有描述符。
0x0104	RC Write Status and Descriptor Base (High)	R/W	指定根复合存储器中写状态基址上 32-bit 及描述符列表。在偏移 0x100 上编程此寄存器下 32 位以前，软件必须编程此寄存器。
0x0108	EP Write Status and Descriptor FIFO Base (Low)	RW	在端点存储器写描述符列表中指定基址的下 32-bit。Write DMA 描述符控制器向根复合存储器请求描述符并把描述符写入该位置。此地址必定是通过 Read DMA Avalon-MM 主端口看到的描述符控制器 Write 描述符列表中 Avalon-MM 从端口的 Avalon-MM 地址。编程偏移 0x10C 的上 32-bit 寄存器后必须编程此寄存器。
0x010C	EP Write Status and Descriptor FIFO Base (High)	RW	指定端点存储器中写描述符列表基址上 32 位。读 DMA 从根复合存储器取回列表并把列表写入该位置。在偏移 0x108 上编程此寄存器下 32-bit 以前，软件必须编程此寄存器。
0x0110	WR_DMA_LAST_PTR	RW	读取时，返回所请求的最后一个描述符 ID。如果无未处理的 DMA 请求或者 DMA 处于复位中，就返回一个值 0xFF。 写入时，被写入时，指定所请求最后一个描述符的 ID。读取值和写入值的不同在于待处理的描述符个数 例如，如果读取值为 4，所请求的最后一个描述符就为 4。要多指定 5 个描述符，软件应该把 9 写入 RD_DMA_LAST_PTR 寄存器。DMA 就多执行 5 个描述符。 要获得读 DMA 记录中每个描述符的 done 位，编程此寄存器每次仅传输一个描述符。 到达 WR_TABLE_SIZE 后，Descriptor ID 环回到 0。 例如，如果 RD_TABLE_SIZE 值读为 126，且您想再多执行 3 个描述符，软件必须写 127，然后写 1 到 RD_DMA_LAST_PTR 寄存器中。
0x0114	WR_TABLE_SIZE	RW	指定 Read 描述符列表的大小。根据描述符的代码设置-1。默认情况下，RD_TABLE_SIZE 被设置为 127。该值指定最后一个 Descriptor ID。要更改 RC Read Status and Descriptor Base (Low) 基地址，必须用完 RD_TABLE_SIZE 指定的所有描述符。
0x0118	WR_CONTROL	RW	[31:1]已保留。 [0]Done。设置后，设置后，描述符控制器为状态列表里的每个描述符写 Done 位。完成最后一个描述符后，描述符控制器发送单个 MSI 中断。当最后一个描述符没有生成 Done 时，就由 WR_DMA_LAST_PTR 指定。

6.11.3. 读 DMA 和写 DMA 描述符列表格式

主机存储器中，读，写描述符储存在各自的描述符列表。每个列表最多储存 128 个描述符。每个描述符为 8 dword，或者 32bytes。读 DMA 和写 DMA 描述符列表开始于 RC Read Descriptor Base 和 RC Write Descriptor Base 地址寄存器中编程地址的 0x200 字节偏移处。

注意：因为 DMA 描述符控制器使用 FIFO 储存描述符列表条目，所以一旦 DMA 描述符控制器开始描述符列表中的指定传输，就不可将它重新编程。

表 65. 读描述符列表格式

地址偏移	寄存器名称	说明
0x00	RD_RC_LOW_SRC_ADDR	读 DMA 源地址的下 dword。在读 DMA 取回数据的根复合存储器中指定地址。
0x04	RD_RC_HIGH_SRC_ADDR	读 DMA 源地址的上 dword。在读 DMA 取回数据的根复合存储器中指定地址。
0x08	RD_CTLR_LOW_DEST_ADDR	在读 DMA 写数据的 Avalon-MM 域中指定地址。
0x0C	RD_CTRL_HIGH_DEST_ADDR	读 DMA 目的地址的上 dword。在读 DMA 写数据的 Avalon-MM 域中指定地址。
0x10	CONTROL	指定以下信息： <ul style="list-style-type: none"> • [31:25] 保留必须为 0。 • [24:18] ID。指定 Descriptor ID。Descriptor ID 0 在列表的前端。Descriptor ID 在列表的末端。 • [17:0] SIZE。传输量以 dword 为单位。且必须为非零。最大传输量为 (1 MB - 4 bytes)。如果所指定传输量小于最大传输量，则传输量为实际输入的大小。
0x14 - 0x1C	保留	N/A

表 66. 写描述符列表格式

地址偏移	寄存器名称	说明
0x00	WR_RC_LOW_SRC_ADDR	写 DMA 源地址的下 dword。在写 DMA 取回数据的 Avalon-MM 域中指定地址。
0x04	WR_RC_HIGH_SRC_ADDR	写 DMA 源地址的上 dword。在写 DMA 取回数据的 Avalon-MM 域中指定地址。
0x08	WR_CTLR_LOW_DEST_ADDR	写 DMA 目的地址的下 dword。在写 DMA 写数据的根复合存储器中指定地址。
0x0C	WR_CTRL_HIGH_DEST_ADDR	写 DMA 目的地址的上 dword。在写 DMA 写数据的根复合存储器中指定地址。
0x10	CONTROL	指定以下信息： <ul style="list-style-type: none"> • [31:25]：已保留必须为 0。 • [24:18]:ID: 指定 Descriptor ID。Descriptor ID 0 在列表前端。Descriptor ID 在列表末端。 • [17:0] :SIZE: 传输量以 dword 为单位。且必须为非零。最大传输量为 (1 MB - 4 bytes)。如果所指定传输量小于最大传输量，则传输量为实际输入的大小。
0x14 - 0x1C	保留	N/A

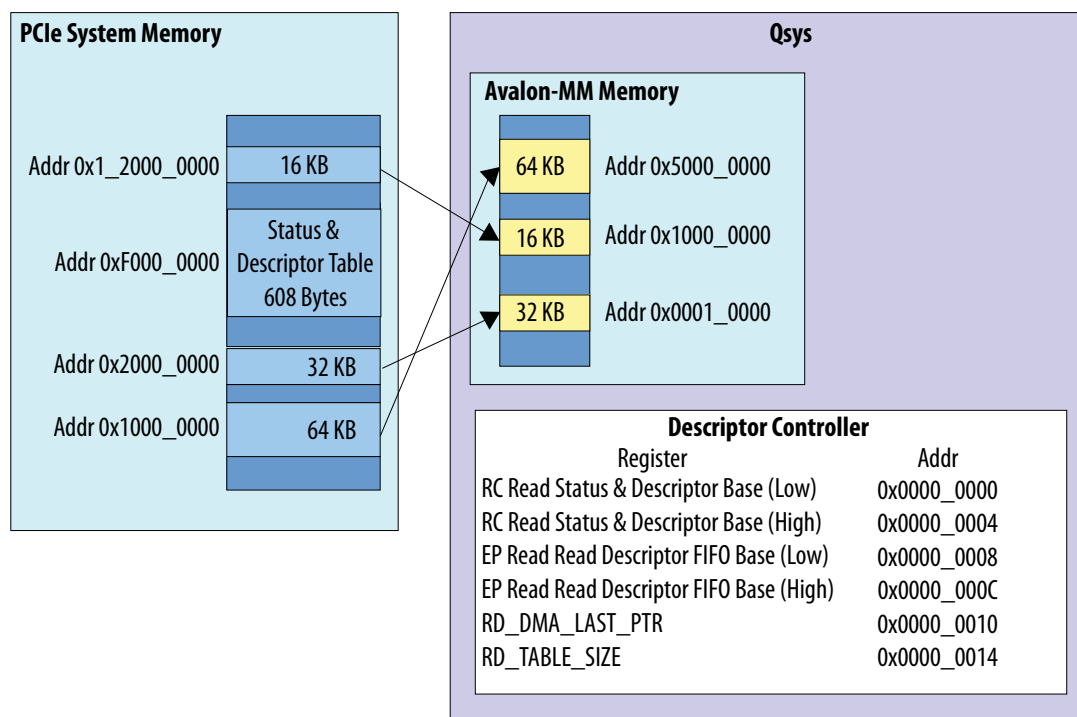
6.11.4. 读 DMA 实例

本实例将 3 个数据块从系统存储器移动到 Avalon-MM address space. 地址空间。在嵌入式 CPU 上运行的主机软件分配存储器并在系统存储器中创建描述符列表。

本实例使用的地址位于 `<install_dir>/ ip/altera/altera_pcie/ altera_pcie_<dev>_ed/example_design/<dev>` 目录的 Qsys 设计实例, **ep_g3x8_avmm256_integrated.qsys** 中。

下图介绍 PCIe 和 Avalon-MM 地址空间中数据块的位置和大小, 以及描述符列表格式。此实例中, RD_TABLE_SIZE 的值为 127。

图 37. 通过读 DMA, 把数据块从 PCIe 地址空间传输到 Avalon-MM 地址空间。



假设描述符列表包含 128 个条目。状态列表优先于存储器中各种数量的描述符。读和写状态, 以及描述符列表分别位于 RC Read Descriptor Base Register 和 RC Write Descriptor Base Register 指定的地址。

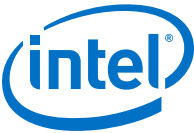
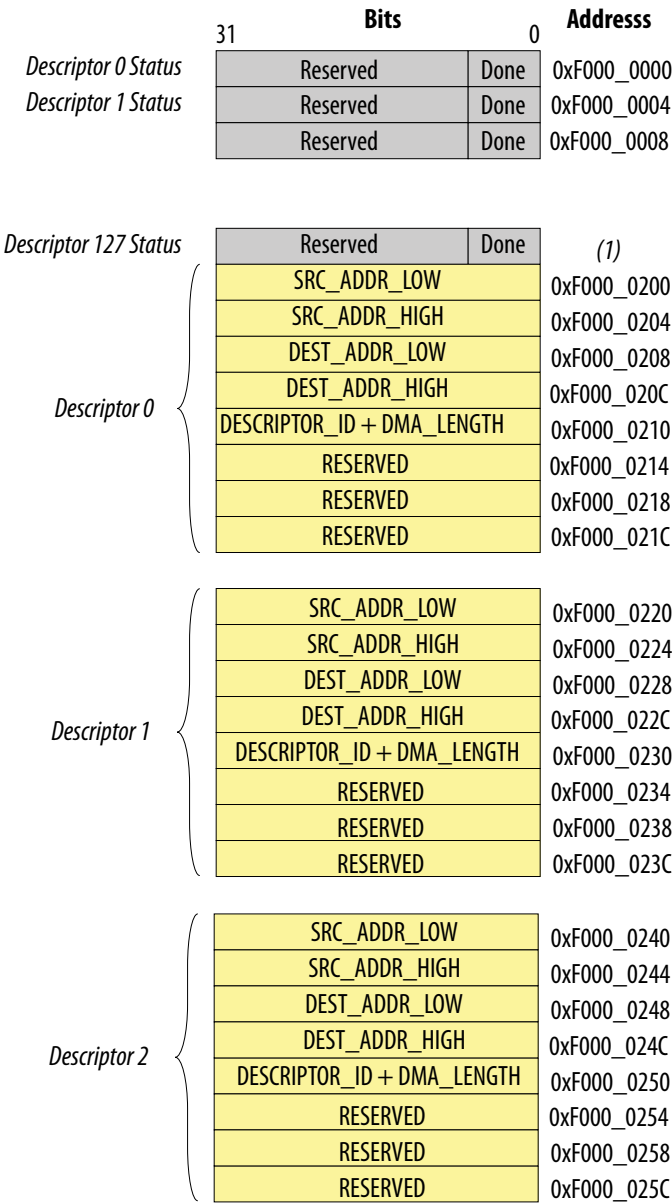


图 38. 描述符列表格式



注释:
1.软件自动添加 0x200 到状态列表的基地址以确认
首个描述符的地址。

- 1. 计算所需的存储器分配：
 - a. 状态列表中每个条目是 4 bytes。128 个条目需要存储器的 512 bytes。
 - b. 每个描述符是 32 bytes。3 个描述符需要存储器的 96 bytes。状态和描述符列表共需分配存储器 608 bytes。
- 2. 在 PCI Express 地址空间中分配存储器的 608 bytes。

此实例中所分配的存储器起始地址为 `0xF000_0000`。将此地址编程到根复合读状态以及描述符基地址寄存器中。

3. 在 **PCI Express** 地址空间创建描述符列表。由于状态列表先于描述符被储存，于是首个描述符被储存在：`0xF000_0000 + 0x200 = 0xF000_0200`。
 - a. 编程 `0x0000_0000` 到描述符 0 的源地址 `0xF000_0204` 中。
这是源地址的上 32 bits。
 - b. 编程 `0x1000_0000` 到描述符 0 的源地址 `0xF000_0200` 中。
这是源地址的下 32 bits。
 - c. 编程 `0x0000_0000` 到描述符 0 的目的地址 `0xF000_020C` 中。
这是目的地址的上 32 bits。
 - d. 编程 `0x5000_0000` 到描述符 0 的目的地址 `0xF000_0208` 中。
这是目的地址的下 32 bits。
通过这四个步骤把存储器 64KB 块的 Avalon-MM 目的地址编程到描述符列表。
 - e. 编程 `0x0000_4000` 到 `0xF000_0210` 以传输描述符 ID 0 数据的 16K dword (64 KB)。
 4. 为第二数据块重复此流程：
 - a. 编程 `0x0000_0000` 到源地址 `0xF000_0224`。
 - b. 编程 `0x2000_0000` 到源地址 `0xF000_0220`。
 - c. 编程 `0x0000_0000` 到目的地址 `0xF000_022C`。
 - d. 编程 `0x0001_0000` 到目的地址 `0xF000_0228`。
 - e. 编程 `0x0004_2000` 到 `0xF000_0230` 以传输描述符 ID 1 数据的 8K dword (32 KB)。
 5. 为第三数据块重复此流程：
 - a. 编程 `0x0000_0001` 到源地址 `0xF000_0244`。
 - b. 编程 `0x2000_0000` 到源地址 `0xF000_0240`。
 - c. 编程 `0x0000_0000` 到目的地址 `0xF000_024C`。
 - d. 编程 `0x1000_00 00` 到目的地址 `0xF000_0248`。
 - e. 编程 `0x0008_1000` 到 `0xF000_0250` 以传输描述符 ID 2 数据的 4K dword (16 KB)。
- 下图显示了完成编程后描述符列表中的值。

图 39. 描述符列表格式

	31	Bits	0	Address
Descriptor 0 Status		Reserved	Done	0xF000_0000
Descriptor 1 Status		Reserved	Done	0xF000_0004
		Reserved	Done	0xF000_0008
Descriptor 127 Status		Reserved	Done	(1)
Descriptor 0	{	1000_0000		0xF000_0200
		0000_0000		0xF000_0204
		5000_0000		0xF000_0208
		0000_0000		0xF000_020C
		0000_4000		0xF000_0210
		RESERVED		0xF000_0214
		RESERVED		0xF000_0218
		RESERVED		0xF000_021C
Descriptor 1	{	2000_0000		0xF000_0220
		0000_0000		0xF000_0224
		0001_0000		0xF000_0228
		0000_0000		0xF000_022C
		0004_2000		0xF000_0230
		RESERVED		0xF000_0234
		RESERVED		0xF000_0238
		RESERVED		0xF000_023C
Descriptor 2	{	2000_0000		0xF000_0240
		0000_0001		0xF000_0244
		1000_0000		0xF000_0248
		0000_0000		0xF000_024C
		0008_1000		0xF000_0250
		RESERVED		0xF000_0254
		RESERVED		0xF000_0258
		RESERVED		0xF000_025C

注释:

1. DMA描述符控制器自动添加0x200到状态列表的基地址以确认首个读描述符的地址。

6. 使用 PCI Express 系统存储器地址空间状态地址和描述符列表地址编程 DMA 描述符控制器。如果是内部 DMA 描述符，就通过组合 BAR0 和 BAR1 访问这些寄存器，因为此实例使用 64-bit 地址。DMA 读控制寄存器起始于偏移 0x0000。写 DMA 控制寄存器起始于偏移 0x0100。
 - a. 编程 0x0000_0000 到偏移 0x0000_0004。
此为存储状态和描述符列表的 PCIe 系统存储器的上 32 bits。

- b. 编程 0xF000_0000 到偏移 0x0000_0000。
此为存储状态和描述符列表的 PCIe 存储器地址的下 32 bits。读 DMA 自动添加一个 0x200 的偏移到该值，以开启基于存储器状态列表的描述符副本。
7. 通过片上 FIFO 地址编程 DMA 描述符控制器。描述符控制器将从该地址复制状态和描述符列表。
 - a. 编程 0x0000_0000 到偏移 0x0000_000C
此为 Avalon-MM 地址域中片上 FIFO 地址的上 32 bits。
 - b. 编程 0x0100_0000 到偏移 0x0000_0008。
此为片上 FIFO 地址的下 32 位。它是内部片上 FIFO 的地址，同时也是透过 RX Master 所见到描述符控制器的一部分。

图 40. 片上 FIFO 的地址

Use	Connections	Name	Description	Base
<input checked="" type="checkbox"/>		DUT	Arria 10 Hard IP for PCI Express	
		txs	Avalon Memory Mapped Slave	0x0000_0000_0000_0000
		rxm_bar4	Avalon Memory Mapped Master	
		dma_rd_master	Avalon Memory Mapped Master	
		dma_wr_master	Avalon Memory Mapped Master	
		rd_dts_slave	Avalon Memory Mapped Slave	0x0000_0000_0100_0000
		wr_dts_slave	Avalon Memory Mapped Slave	0x0000_0000_0100_2000
		rd_dcm_master	Avalon Memory Mapped Master	
		wr_dcm_master	Avalon Memory Mapped Master	
<input checked="" type="checkbox"/>		onchip_memory2_0	On-Chip Memory (RAM or ROM)	
		s1	Avalon Memory Mapped Slave	0x0000_0000_0000_0000
		s2	Avalon Memory Mapped Slave	0x0000_0000_0000_0000

8. 编程描述符控制器 RD_DMA_LAST_PTR 寄存器。
由此步骤开启 DMA。完成 3 个描述符后也由它指定需要被更新的状态 dword。
 - 将最后一个描述符单独更新为 done 位，并编程 0x2 到偏移 0x0000_0010。描述符控制器处理完全部 3 个描述符并在状态列表中的 0xF000_0008 写 done 位。
 - 要把全部 3 个描述符更新为 done 位，使用值 0, 1, 2 三次编程地址 0x0000_0010 RD_DMA_LAST_PTR。描述符控制器为地址 0xF000_0000, 0xF000_0004, 及 0xF000_0008 设置 done 位。如果系统支持无序读完成，则描述符控制器可能会无序完成描述符。在这样的系统中，您必须采用这样的方法为每个描述符请求 done 状态。软件必须查看每个描述符的 done 状态。

相关链接

了解内部 DMA 描述符控制器 (第 98 页)

6.11.5. 软件编程以同时读和写 DMA

按以下步骤编程从而实现按 DMA 同步传输：

1. 为读和写 DMA 描述符列表分配根端口存储器。假设列表包含多达 128 个，8-dword 描述符和 128 个，1-dword 状态条目，总共 1152 个 dword。用于读和写 DMA 描述符列表的存储器总容量为 2304 dword。
2. 分配根端口存储器并通过数据将它初始化以用于读 DMA 进行读取。
3. 分配根端口存储器以用于写 DMA 写入。



4. 为读 DMA 描述符列表创建所有描述符。逐步分配 DMA Descriptor ID, 从 0 开始, 最多到 127。对于读 DMA 来说, 源地址是步骤 2 中所分配的存储器空间。目的地址是读 DMA 模块写入的 Avalon-MM 地址。以 dword 为单位指定 DMA 长度。每个描述符传输相邻的存储器。假设用于读 DMA 的基地址为 0, 以下分配说明了一个读描述符的构成:
 - a. RD_RC_LOW_SRC_ADDR = 0x0000 (根端口中用于读描述符列表的基地址)
 - b. RD_RC_HIGH_SRC_ADDR = 0x0004
 - c. RD_CTLR_LOW_DEST_ADDR 0x0008
 - d. RD_CTLR_HIGH_DEST_ADDR = 0x000C
 - e. RD_DMA_LAST_PTR = 0x0010
 写 RD_DMA_LAST_PTR 寄存器开始操作。
5. 关于写 DMA, 源地址是写 DMA 模块应该读取的 Avalon-MM 地址。目的地址是步骤 3 中所分配的根端口存储器空间。以 dword 为单位指定 DMA 长度。假设用于写 DMA 的基地址为 0x100, 以下分配说明一个写描述符的构成:
 - a. RD_RC_LOW_SRC_ADDR = 0x0100 (根端口中用于读描述符列表的基地址)
 - b. WD_RC_HIGH_SRC_ADDR = 0x0104
 - c. WD_CTLR_LOW_DEST_ADDR 0x0108
 - d. WD_CTLR_HIGH_DEST_ADDR = 0x010C
 - e. WD_DMA_LAST_PTR = 0x0110
 写 WD_DMA_LAST_PTR 寄存器开始操作。
6. 为提高吞吐量, 读 DMA 模块在操作开始之前就将描述符列表复制到 Avalon-MM 存储器。通过写 EP Descriptor Table Base (Low)和(High)寄存器指定存储器地址。
7. 为每个 WD_DMA_LAST_PTR 或 RD_DMA_LAST_PTR 完成发送一个 MSI 中断。这些完成会引起 done 状态位的更新。然后主机软件会读状态位以决定哪个 DMA 操作已被完成。

注意:

支持读 DMA 请求的无序完成。如果读 DMA 的传输量超过读请求的最大量, 则读 DMA 创建多个读请求。例如, **Maximum Read Request Size** 为 512 bytes, 读 DMA 把一个 4 KB 读请求分成 8 个请求并使用 8 个不同标签标示。读完成可按任何顺序返回。读 DMA Avalon-MM 主端口根据标签把读完成写入正确的位置。

6.12. 控制寄存器访问 (CRA) Avalon-MM 从端口

表 67. 配置空间寄存器说明

可选的 CRA Avalon-MM 从端口提供主机访问到所选配置空间和状态寄存器。这些寄存器为只读。少于 32 bits 的寄存器不使用上位。

字节偏移	寄存器	Dir	说明
14'h0000	cfg_dev_ctrl[15:0]	0	cfg_devctrl[15:0]是 PCI Express 性能结构的器件控制。
14'h0004	cfg_dev_ctrl2[15:0]	0	cfg_dev2ctrl[15:0] 是 PCI Express 性能结构的器件控制 2。
14'h0008	cfg_link_ctrl[15:0]	0	cfg_link_ctrl[15:0]是 PCI Express 性能结构的首要链路控制 (primary Link Control) 。
继续...			



字节偏移	寄存器	Dir	说明
			对于 Gen2 或 Gen3 的操作, 您必须写 1' b1 到根端口的 Retrain Link bit (cfg_link_ctrl 的 Bit[5]) 以初始化再训练 (retraining) 从而在 Gen1 L0 状态的初始化链路训练后达到较高数据速率。再训练指导 LTSSM 进入恢复状态。如果链路上的两个器件都具有较高数据速率性, 再训练以达到较高数据速率就不会是自动事件。
14'h000C	cfg_link_ctrl2[15:0]	O	cfg_link_ctrl2[31:16] 是 Gen2 操作中 PCI Express 性能结构的次要链路控制寄存器。 关于 Gen1 variant, 链路带宽通知位总是设置为 0。而 Gen2 variant, 此位设置为 1。
14'h0010	cfg_prm_cmd[15:0]	O	PCIe 配置空间的基本/主要命令寄存器。
14'h0014	cfg_root_ctrl[7:0]	O	PCI-Express 性能的根控制和状态寄存器。此寄存器仅根端口模式中可用。
14'h0018	cfg_sec_ctrl[15:0]	O	PCI-Express 功能的次要总线控制和状态寄存器。此寄存器仅根端口模式中可用。
14'h001C	cfg_secbus[7:0]	O	次要总线 (Secondary bus) 数目。根端口模式中可用。
14'h0020	cfg_subbus[7:0]	O	附属总线 (Subordinate bus) 数目。根端口模式中可用。
14'h0024	cfg_msi_addr_low[31:0]	O	cfg_msi_add[31:0] 是 MSI 消息地址。
14'h0028	cfg_msi_addr_hi[63:32]	O	cfg_msi_add[63:32] 是 MSI 上层消息地址。
14'h002C	cfg_io_bas[19:0]	O	Type 1 配置空间的 IO 基础寄存器。此寄存器仅根端口模式中可用。
14'h0030	cfg_io_lim[19:0]	O	Type 1 配置空间的 IO 限制寄存器。此寄存器仅根端口模式中可用。
14'h0034	cfg_np_bas[11:0]	O	Type 1 配置空间的非预取存储器基础寄存器。此寄存器仅根端口模式中可用。
14'h0038	cfg_np_lim[11:0]	O	Type 1 配置空间的非预取存储器限制寄存器。此寄存器仅根端口模式中可用。
14'h003C	cfg_pr_bas_low[31:0]	O	Type 1 配置空间可预取基础寄存器的下 32 bits。此寄存器仅根端口模式中可用。
14'h0040	cfg_pr_bas_hi[43:32]	O	Type 1 配置空间可预取基础寄存器的上 12 bits。此寄存器仅根端口模式中可用。
14'h0044	cfg_pr_lim_low[31:0]	O	Type 1 配置空间可预取基础寄存器的下 32 bits。此寄存器仅根端口模式中可用。
14'h0048	cfg_pr_lim_hi[43:32]	O	Type 1 配置空间可预取限制寄存器上 12 bits。此寄存器仅根端口模式中可用。
14'h004C	cfg_pmcsr[31:0]	O	cfg_pmcsr[31:16] 为电源管理控制和 cfg_pmcsr[15:0] 为电源管理状态寄存器。
14'h0050	cfg_msixcsr[15:0]	O	MSI-X 消息控制寄存器。
14'h0054	cfg_msicsr[15:0]	O	MSIMSI 消息控制。
14'h0058	cfg_tvcmap[23:0]	O	配置流量等级 (TC) / 虚拟通道 (VC) 映射。应用层使用此信号生成一个 TLP, 映射基于数据包流量等级的恰当通道。 编码定义如下: <ul style="list-style-type: none"> cfg_tvcmap[2:0]: TC0 的映射 (总为 0)。 cfg_tvcmap[5:3]: TC1 的映射。 cfg_tvcmap[8:6]: TC2 的映射。 cfg_tvcmap[11:9]: TC3 的映射。 cfg_tvcmap[14:12]: TC4 的映射。

继续..

6. 寄存器

UG-01145_avmm_dma | 2016.10.31

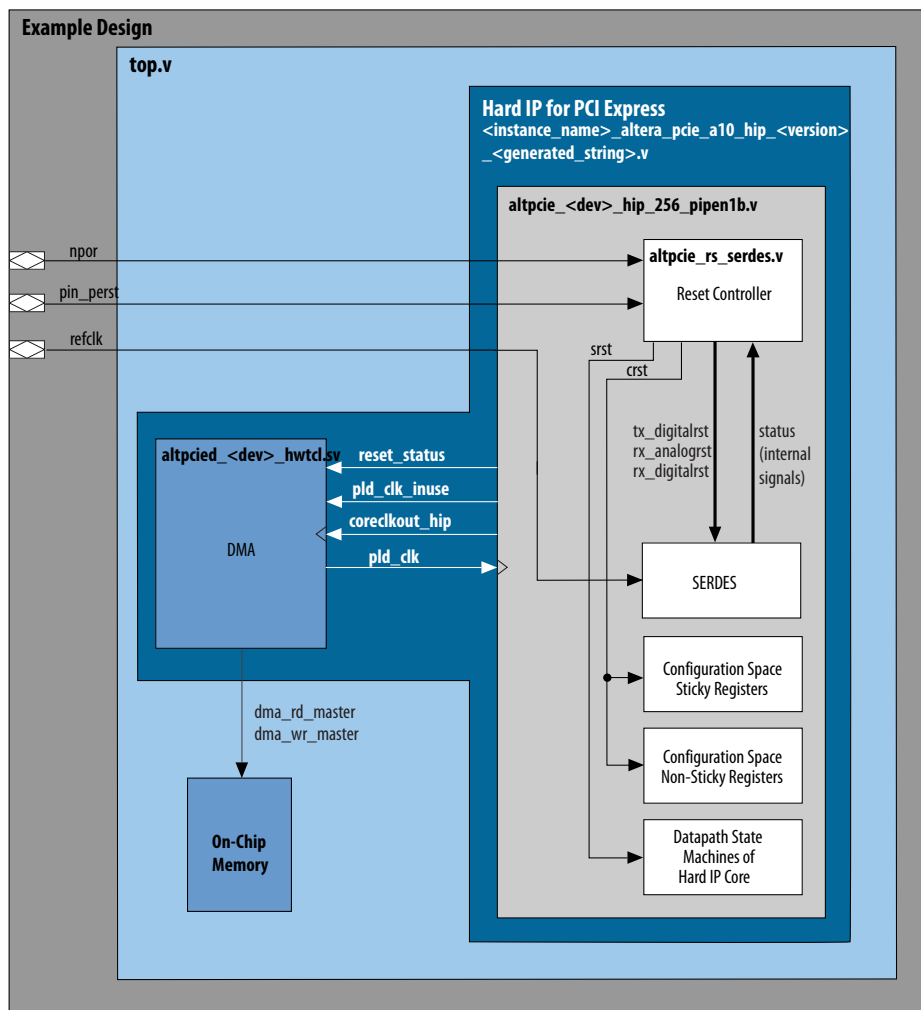


字节偏移	寄存器	Dir	说明
			<ul style="list-style-type: none"> cfg_tcvcmmap[17:15]: TC5 的映射。 cfg_tcvcmmap[20:18]: TC6 的映射。 cfg_tcvcmmap[23:21]: TC7 的映射。
14'h005C	cfg_msi_data[15:0]	0	cfg_msi_data[15:0] 是 MSI 的消息数据。
14'h0060	cfg_busdev[12:0]	0	由 Hard IP 捕捉的总线/器件数或已编程到 Hard IP 的总线/器件数。 域定义如下: <ul style="list-style-type: none"> cfg_busdev[12:5]: 总线数 cfg_busdev[4:0]: : 器件数
继续...			

字节偏移	寄存器	Dir	说明
14'h0064	ltssm_reg[4:0]	O	<p>指定当前 LTSSM 状态。LTSSM 状态机编码定义以下状态：</p> <ul style="list-style-type: none"> 5'b: 00000: Detect.Quiet (检测。静音) 5'b: 00001: Detect.Active (检测。有效) 5'b: 00010: Polling.Active (轮询。有效) 5'b: 00011: Polling.Compliance (轮询。合规) 5'b: 00100: Polling.Configuration (轮询。配置) 5'b: 00101: Polling.Speed (轮询。速度) 5'b: 00110: config.Linkwidthstart (配置。链路宽度开始) 5'b: 00111: Config.Linkaccept (配置。链路接受) 5'b: 01000: Config.Lanenumaccept (配置。通道数接受) 5'b: 01001: Config.Lanenumwait (配置。通道数等待) 5'b: 01010: Config.Complete (配置。完成) 5'b: 01011: Config.Idle (配置。空闲) 5'b: 01100: Recovery.Rcvlock (恢复。恢复时钟) 5'b: 01101: Recovery.Rcvconfig (恢复。恢复配置) 5'b: 01110: Recovery.Idle (恢复。空闲) 5'b: 01111: L0 5'b: 10000: Disable (禁用) 5'b: 10001: Loopback.Entry (环回。进入) 5'b: 10010: Loopback.Active (环回。有效) 5'b: 10011: Loopback.Exit (环回。退出) 5'b: 10100: Hot.Reset (热。复位) 5'b: 10101: L0s 5'b: 11001: L2.transmit.Wake (L2。发送。唤醒) 5'b: 11010: Speed.Recovery (速度。恢复) 5'b: 11011: Recovery.Equalization, Phase 0 (恢复。均衡, Phase 0) 5'b: 11100: Recovery.Equalization, Phase 1 (恢复。均衡, Phase 1) 5'b: 11101: Recovery.Equalization, Phase 2 (恢复。均衡, Phase 2) 5'b: 11110: recovery.Equalization, Phase 3 (恢复。均衡, Phase 3)
14'h0068	current_speed_reg[1:0]	O	<p>显示 PCIe 链路的当前速度。编码定义如下：</p> <ul style="list-style-type: none"> 2b' 00: 未定义 2b' 01: Gen1 2b' 10: Gen2 2b' 11: Gen3
14'h006C	lane_act_reg[3:0]	O	<p>通道有效模式：该信号显示链路训练期间执行配置的通道数。编码定义如下：</p> <ul style="list-style-type: none"> 4' b0001: 1 个 lane 4' b0010: 2 个 lanes 4' b0100: 4 个 lanes 4' b1000: 8 个 lanes

7. Arria 10 复位和时钟

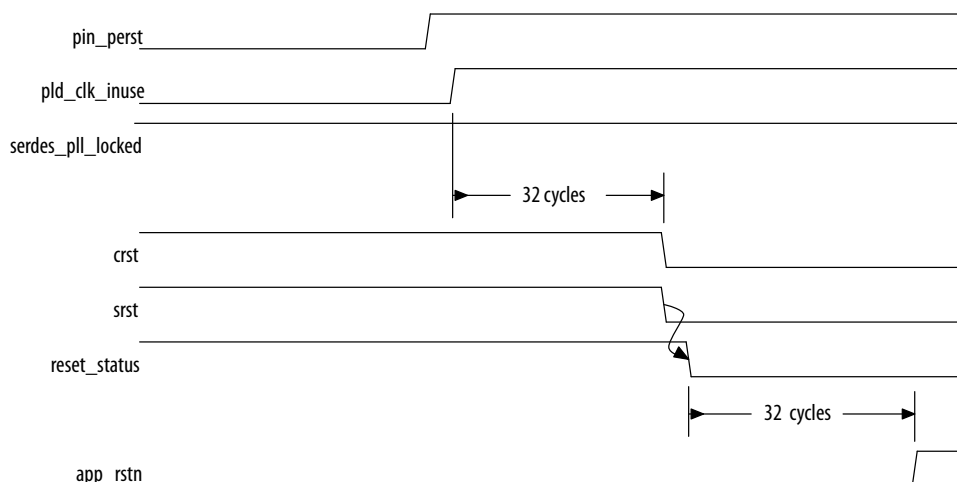
图 41. Arria 10 器件中的复位控制器



7.1. Hard IP for PCI Express IP 内核及应用层复位序列

图 42. Hard IP for PCI Express 及应用逻辑复位序列

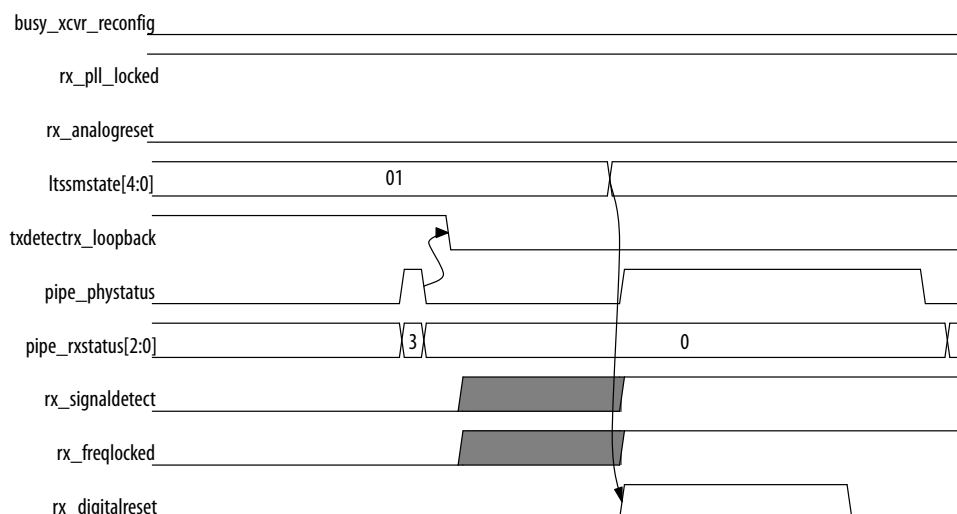
您的应用层可例化一个与图示中相似的模块以生成 `app_rstn`，用来复位应用层逻辑。



该复位序列包含以下步骤：

1. 在释放 `pin_perst` 或 `npwr` 之后，硬核 IP 复位控制器等待 `pld_clk_inuse` 被置位。
2. 置位 `pld_clk_inuse` 后，释放 `crst` 和 `srst` 32 个周期。
3. Hard IP for PCI Express 置低 `reset_status` 输出到应用层。
4. 释放 `reset_status` 后，`altpciied_<device>v_hwtcl.sv` 置低 `app_rstn` 32 个 `pld_clk` 周期。

图 43. RX 收发器复位序列

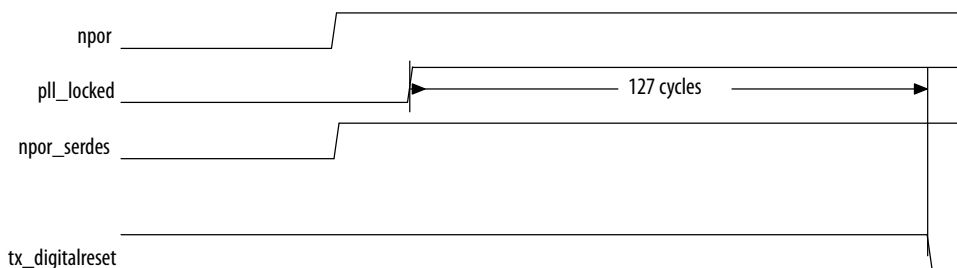




RX 收发器复位序列包含以下步骤:

1. 置位 `rx_pll_locked` 后, 从 Detect.Quiet 到 Detect.Active 状态的 LTSSM 状态机转换。
2. 当置位 `pipe_phystatus` 脉冲且 `pipe_rxstatus[2:0] = 3` 时, 接收器检测操作已完成。
3. 从 Detect.Active 状态到 Polling.Active 状态的 LTSSM 状态机转换。
4. Hard IP for PCI Express 置位 `rx_digitalreset`。 `rx_signaldetect` 最少稳定 3ms 后, `rx_digitalreset` 信号被置低。

图 44. TX 收发器复位序列



TX 收发器复位序列包含以下步骤:

1. `npor` 被置低后, IP 内核置低到 TX 收发器的 `npor_serdes` 输入。
2. SERDES 复位控制器等待 `pll_locked` 至少稳定 127 个 `p1d_clk` 周期, 然后置低 `tx_digitalreset`。

关于可用复位信号的说明, 请参阅 *Reset Signals, Status, and Link Training Signals* (复位信号, 状态, 和链路训练信号)。

7.2. 时钟

硬核 IP 包含一个跨时钟域 (CDC) 同步器, 并位于 PHY/MAC 和 DLL 层的接口。同步器允许数据链路和事务层在独立于 PHY/MAC 频率中运行。CDC 同步器为用户时钟接口提供更多灵活性。基于您指定的参数, 内核会选择正确的 `coreclkout_hip`。可以使用这些参数来提高性能, 如通过在较高频率运行以优化延迟, 或在较低频率运行以节省电耗。

根据 *PCI Express Base Specification*, 您必须提供一个与收发器直接相连的 100 MHz 参考时钟。

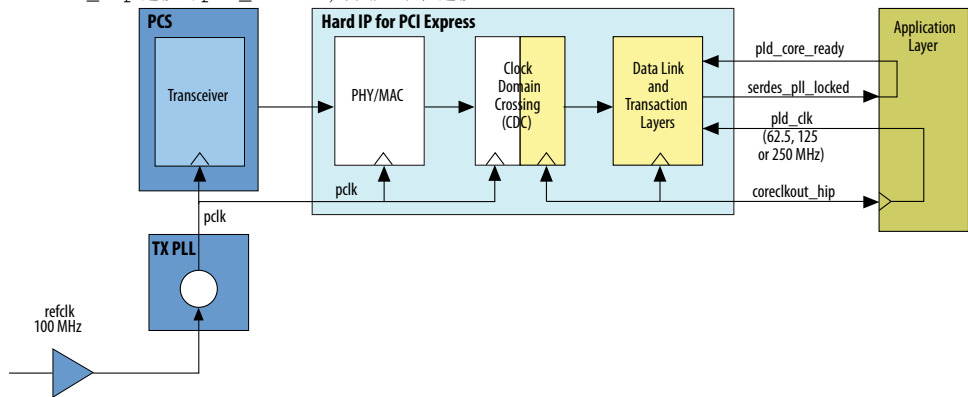
相关链接

[PCI Express Base Specification 3.0](#)

7.2.1. 时钟域

图 45. 应用层的时钟域和时钟生成

下图说明使用 coreclkout_hip 驱动应用层及 IP 内核 pld_clk 时的时钟域。Intel 所提供的设计实例是将 coreclkout_hip 连接到 pld_clk。但，并非强制性连接。



如图所示，IP 内核包含以下时钟域：

7.2.1.1. coreclkout_hip

表 68. 用于链路宽度，数据速率和应用层接口宽度间所有组合的应用层时钟频率。

coreclkout_hip 信号来自于 pclk。以下表格罗列了用于 coreclkout_hip 的频率，也是应用层到事务层接口的链路宽度，数据速率以及宽度的函数。整个操作中都维持本表格中指定的频率和宽度。如果链路下行到一个较少链路宽度或改变到一个不同的最大链路率，也继续保持本列表中指定的初始配置使用的频率。（硬核 IP 调节接口以达到较低吞吐量。）

链路宽度	最大链路速率	Avalon 接口宽度	coreclkout_hip
×8	Gen1	128	125 MHz
×4	Gen2	128	125 MHz
×8	Gen2	128	250 MHz
×8	Gen2	256	125 MHz
×2	Gen3	128	125 MHz
×4	Gen3	128	250 MHz
×4	Gen3	256	125 MHz
×8	Gen3	256	250 MHz

7.2.1.2. pld_clk

coreclkout_hip 可通过 pld_clk 输入单独驱动应用层时钟到 IP 内核。pld_clk 可作为 coreclkout_hip 以外其他时钟的可选时钟源。pld_clk 的最低频不可低于 coreclkout_hip 的频率。基于指定应用层的约束，可使用 PLL 来产生所需频率。



7.2.2. 时钟摘要

表 69. 时钟摘要

名称	频率	时钟域
coreclkout_hip	62.5, 125 或 250 MHz	事务层和应用层间的 Avalon-ST 接口。
pld_clk	125 或 250 MHz	应用层和事务层。
refclk	100 MHz	SERDES (收发器)。专用自由运行输入时钟到 SERDES 块。



8. 错误处理

每个兼容 PCI Express 的器件都必须实现一个基础级的错误管理并选择性实现高级错误管理。IP 内核实现基础和高级报错。根端口的错误处理比端点的要更复杂。

表 70. 错误分类

PCI Express Base Specification 定义三种错误类型，如下表格中概述。

类型	相关代理	说明
可纠正	硬件	当可纠正的错误可能影响到系统性能时，仍保留数据完整性。
不可纠正，不严重	器件软件	不可纠正，不严重错误的定义是数据丢失，但保持系统完整性的错误。例如，架构可能丢失了某个特定的 TLP，但仍可正常运行。
不可纠正，严重	系统软件	由于丢失数据和系统故障而产生的错误，被认为是不可纠正且严重的。软件必须决定如何处理这样的错误：是否复位链路还是实现其他途径将问题最小化。

[相关链接](#)

[PCI Express Base Specification 3.0](#)

8.1. 物理层错误

表 71. 物理层检测到的错误

以下列表介绍了由物理层检测到的错误。物理层报错在 PCI Express Base Specification 中为可选。

错误	类型	说明
接收端口错误	可纠正	<p>造成错误的 3 种潜在原因如下：</p> <ul style="list-style-type: none"> 当 lane 处于 L0 状态时，出现物理编码子层错误。通过每 lane PIPE 接口输入接收的状态信号，<code>rxstatus<lane_number>[2:0]</code>，把这些错误报告到硬核 IP 块。此状态信号使用下列编码： <ul style="list-style-type: none"> 3'b100: 8B/10B 解码错误 3'b101: 弹性缓冲上溢 3'b110: 弹性缓冲下溢 3'b111: 差异错误 多通道去偏移 FIFO 上溢导致的去偏移错误。 控制符被错误的 lane 接收。

8.2. 数据链路层错误

表 72. 数据链路层检测到的错误

错误	类型	说明
Bad TLP	可纠正	该错误出现在 LCRC 验证失败或序列号出错时。
Bad DLLP	可纠正	该错误出现在 CRC 验证失败时。
继续...		

Intel Corporation. All rights reserved. Agilix, Altera, Arria, Cyclone, Enpirion, Intel, the Intel logo, MAX, Nios, Quartus and Stratix words and logos are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

*Other names and brands may be claimed as the property of others.



错误	类型	说明
Replay timer	可纠正	重放定时器超时，就出现该错误。
Replay num rollover	可纠正	该错误出现在重放编号翻转时。
Data Link Layer protocol	不可纠正（严重）	该错误出现在数据链路层中 Ack/Nak 块指定的序列号 (AckNak_Seq_Num) 与未确认 TLP 不一致时。

8.3. 事务层错误

表 73. 事务层检测到的错误

错误	类型	说明
Poisoned TLP received	不可纠正（非严重）	如果已接收事务层数据包具有 EP 中毒位设置，则该错误出现。 已接收 TLP 传递到应用层，应用层逻辑必须采取正确行动回应已中毒 TLP。请参阅 <i>PCI Express Base Specification</i> 中“2.7.2.2 Rules for Use of Data Poisoning”部分了解更多有关中毒 TLP 的信息。
ECRC check failed ⁽¹⁾	不可纠正（非严重）	该错误由 ECRC 检查失败导致，尽管 TLP 不是畸形并且 LCRC 检查有效。 硬核 IP 块自动处理此 TLP。如果此 TLP 为 non-posted 请求，硬核 IP 块生成一个完成器中止状态的完成。任何情况下，硬核 IP 中删除 TLP 且未在申请层出现。
Unsupported Request for Endpoints	不可纠正（非严重）	每当组件接收到下列任何不支持的请求时，就出现该错误： <ul style="list-style-type: none"> Type 0 配置请求的功能不存在。 请求器 ID 不匹配总线，器件和功能数的完成交易。 不支持的消息。 关于 PCIe 链路中 TLP 的 Type 1 配置请求。 本机端点上的锁定存储器读 (MEMRDLC)。 锁定的完成交易。 地址的 32 个 MSB 被设置为 0 的 64-bit 存储器交易。 没有 BAR 匹配的存储器或 I/O 交易。 Memory Space Enable 位（在配置空间偏移 0x4 的 PCI 命令寄存器的位[1]）设置为 0 时的存储器交易。 中毒的配置写请求 (CfgWr0) 任何情况下，TLP 在硬核 IP 中被删除以及未出现于应用层。如果 TLP 为 non-posted 请求，硬核 IP 块生成一个不支持请求状态的完成。
Completion timeout	不可纠正（非严重）	此错误出现在从应用层发起的请求未在建立期间生成相应的完成 TLP 时。 应用层逻辑负责提供完成超时机制。由事务层 cpl_err[0] 信号报告此完成超时。
Completer abort ⁽¹⁾	不可纠正（非严重）	当中止接收 TLP 时，应用层使用 cpl_err[2] 信号报告此错误。
Unexpected completion	不可纠正（非严重）	此错误由意外的完成交易引起。Hard IP 块处理以下情况： <ul style="list-style-type: none"> 完成数据包中的请求器 ID 与端点的已配置 ID 不匹配。 完成数据包有一个无效的标签代码。（通常，完成数据包中使用的标签超出指定标签的数量） 完成数据包有一个与未处理请求不匹配的标签。 到 I/O 或配置空间的请求完成包比 1 个 dword 长。 响应未传输到配置空间的请求时，Configuration Retry Status (CRS) 就是完成状态。 以上所有情况中，TLP 不出现在应用层；硬核 IP 块将其删除。 应用层通过 cpl_err[2] 信号检测和报告其他意外完成情况。例如，应用层报告接收到成功完成的长度与读请求原本长度不符。
Receiver overflow ⁽¹⁾	不可纠正（严重）	此错误出现在组件接收的 TLP 违反分配到此类 TLP 的 FC 信用时。任何情况下，硬核 IP 中删除 TLP 且不传输到应用层。

继续...

错误	类型	说明
Flow control protocol error (FCPE) ⁽¹⁾	不可纠正 (严重)	此错误出现于组件未在 200 μ s 限时中接收更新流程控制信用时。
Malformed TLP	不可纠正 (严重)	<p>该错误可由以下任何一种情况导致：</p> <ul style="list-style-type: none"> 所接收 TLP 的数据负载超出最大负载量。 TD 域被置位但不存在 TLP digest，或存在 TLP digest 但 PCI Express 请求头数据包的 TD 位没有被置位。 TLP 违反字节使能规则。硬核 IP 块查看该违规，PCI Express specification 将此视为可选。 TLP 中 type 和 length 域与 TLP 总长度不一致。 TLP 中格式和类型的组合不是由 PCI Express specification 所指定。 请求指定的地址/长度组合导致存储器空间访问超出 4 KByte 边界的。PCI Express 规范将硬核 IP 块检查此违规作为可选。 消息（如，Assert_INTX，Power Management，Error Signaling，Unlock 和 Set Power Slot Limit）必须在默认流量级别中发送。 <p>硬核 IP 块删除畸形 TLP；且不出现在应用层。</p>
注释： 1. PCI Express Base Specification Revision 中作为可选。		

8.4. 错误报告和数据中毒

端点如何处理一个特定的错误取决于器件的配置寄存器。

请参阅 *PCI Express Base Specification 3.0* 了解关于器件的信号发送和端点日志记录的说明。

硬核 IP 块实现数据中毒是一种用于表示与交易相关的数据已损坏的机制。中毒的 TLP 具有设置为 1 的头错误/中毒位，并遵守以下规则：

- 接收的中毒 TLP 被发送到应用层且状态位在配置空间被自动更新。
- 接收的已中毒 Configuration Write TLP 不被写入配置空间。
- 配置空间从不生成中毒的 TLP；头的错误/中毒位总是设置为 0。

中毒的 TLP 也可设置 PCI 配置空间状态寄存器中的奇偶校验错误位。

表 74. 奇偶校验错误条件

状态位	条件
Detected parity error (status register bit 15)	接收到任何中毒 TLP 时设置。
Master data parity error (status register bit 8)	<p>当命令寄存器奇偶校验使能位被设置，以及满足任何下列条件时，就设置此位：</p> <ul style="list-style-type: none"> 写请求 TLP 发送期间设置了中毒位。 所接收的完成 TLP 上设置了中毒位。

Hard IP 块接收的中毒数据包被传递到应用层。中毒的发送 TLP 也被类似地发送到链路。

[相关链接](#)

[PCI Express Base Specification 3.0](#)

8.5. 不可纠正和可纠正的错误状态位

以下部分经 PCI-SIG 许可而转载。Copyright 2010 PCI-SIG。

图 46. 不可纠正的错误状态寄存器

该寄存器中所有位的默认值为 0。设置一个错误状态位表示该位所代表的错误情况已被检测到。软件可能通过写 1 到适当的位以清除错误状态。

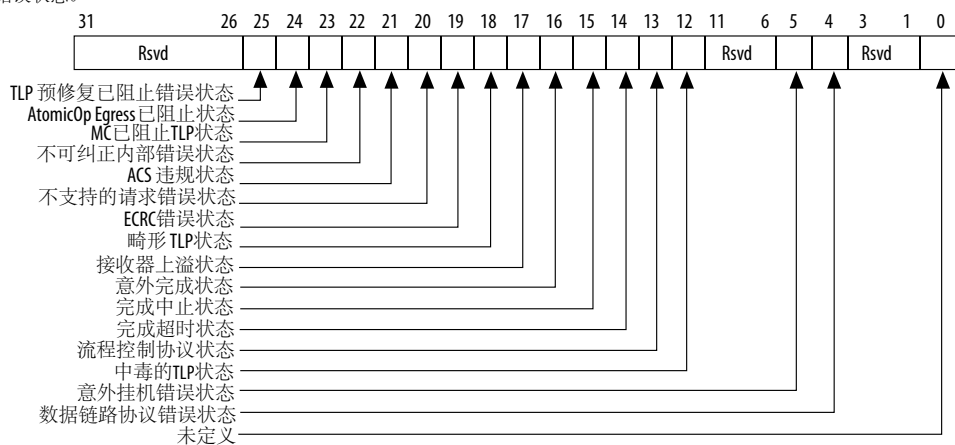
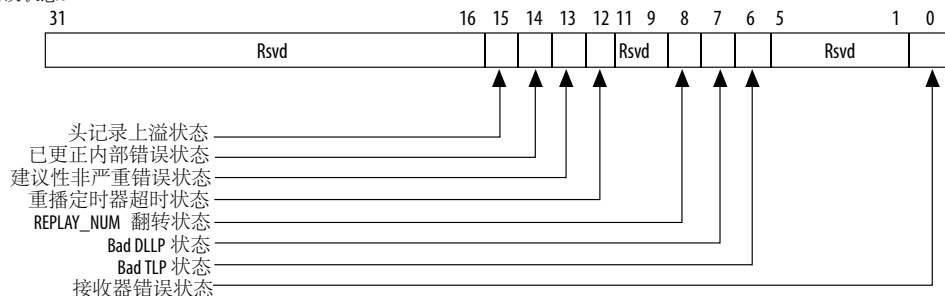


图 47. 可纠正错误状态寄存器

该寄存器中所有位的默认值为 0。设置一个错误状态位表示该位所代表的错误情况已被检测到。软件可能通过写 1 到适当的位以清除错误状态。



9. IP Core 体系结构

Arria10 Avalon-MM Hard IP for PCI Express 按照 *PCI Express Base Specification* 的定义实现完成 PCI Express 协议栈。协议栈包括以下分层：

- **事务层**—事务层包含配置空间，也管理应用层，RX 和 TX 通道，RX 缓冲器，以及流程控制信用。
- **数据链路层**—数据链路层，位于物理层和事务层之间，管理数据包发送和维护链路级的数据完整性。尤其，具体而言，数据链路层执行以下任务：
 - 管理数据链路层数据包（DLLPs）的发送和接收
 - 生成所有发送循环冗余代码（CRC）值并在接收期间查看所有的 CRC
 - 根据所接收的 ACK/NAK 数据链路层数据包管理重试缓冲器和重试机制
 - 初始化用于 DLLPs 的流程控制机制并路由流程控制信用流入和流出事务层
- **物理层**—物理层根据从链路接收到的数据包初始化速率，lane 编号，和 PCI Express 链路的 lane 宽度，并管理并管理收到来自较高层接的数据包。

下图为高级别框图。

图 48. Arria10 Avalon-MM DMA for PCI Express

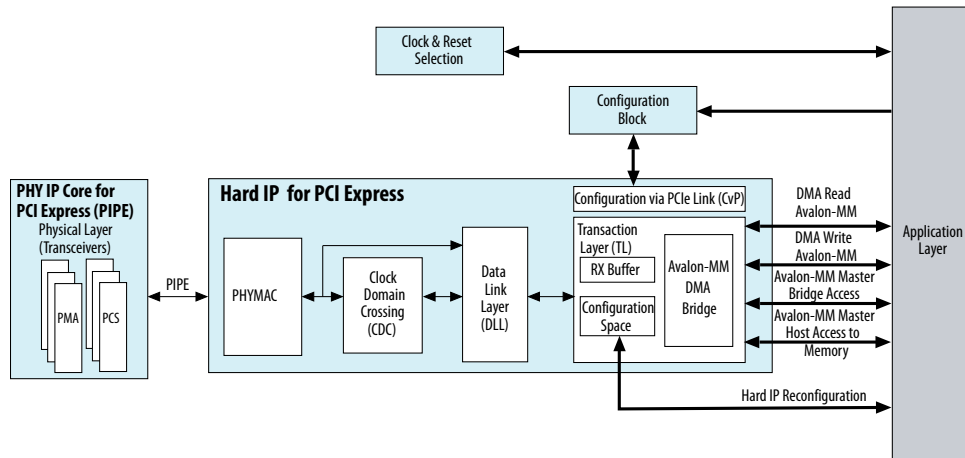


表 75. 应用层时钟频率

Lane	Gen1	Gen2	Gen3
×2	N/A	N/A	125 MHz @ 128 bits
×4	N/A	125 MHz @ 128 bits	250 MHz @ 128 bits 或

继续...



Lane	Gen1	Gen2	Gen3
			125 MHz @ 256 bits
x8	125 MHz @ 128 bits	250 MHz @ 128 bits 或 125 MHz @ 256 bits	250 MHz @ 256 bits

[相关链接](#)

[PCI Express Base Specification 3.0](#)

9.1. 顶层接口

9.1.1. Avalon-MM DMA 接口

具有 DMA 的 Avalon-MM 接口把应用层和事务层连接起来。此接口包含高性能，可突发 Read DMA 和 Write DMA 模块。该 variant 可用于以下端点配置：

- Gen1 x8
- Gen2 x4, x8
- Gen3 x2, x4, x8

[相关链接](#)

[Arria10 DMA Avalon-MM DMA 接口到应用层 \(第 37 页\)](#)

9.1.2. 时钟和复位

PCI Express Base Specification 需要一个输入参考时钟，本设计中称为 `refclk`。*PCI Express Base Specification* 规定该时钟的频率为 100 MHz。

PCI Express Base Specification 还需要 100 ms 的系统配置时间。为符合此规范，IP 内核包含一个嵌入的硬核复位控制器。此复位控制器在器件外设初始化后就退出复位状态。

9.1.3. 中断

Hard IP for PCI Express 提供以下中断机制：

- **Message Signaled Interrupts (MSI)** — MSI 使用 TLP 单 dword 存储器写来实现中断。该中断机制能节省管脚数目，因为此中断不使用单独的线路。另外，单个 dword 为中断消息中出现的数据提供灵活性。MSI Capability 结构储存在配置空间，并通过配置空间访问对其编程。
- **MSI-X**—事务层生成由单个 dword 存储器写的 MSI-X 消息。MSI-X Capability 结构指向保存在存储器的 MSI-X Table 结构和 MSI-X PBA 结构。该方案与 MSI capability 结构相反，它具有关于中断矢量的全部控制和状态信息。

[相关链接](#)

[端点的 MSI 中断 \(第 50 页\)](#)

9.1.4. PIPE

PIPE 接口实现 Intel 设计的 PIPE 接口规范。可使用此平行接口加速仿真；然而，不可在实际硬件中使用 PIPE 接口。

- Gen1, Gen2, 和 Gen3 仿真模型支持 PIPE 和串型仿真。
- 关于 Gen3, Intel 引导端口 BFM 旁路 Gen3 阶段 2 和阶段 3 均衡。然而，如果在第三方 BFM 指导下 Gen3 variant 可执行阶段 2 和阶段 3 均衡。

相关链接

[PIPE 接口信号](#) (第 53 页)

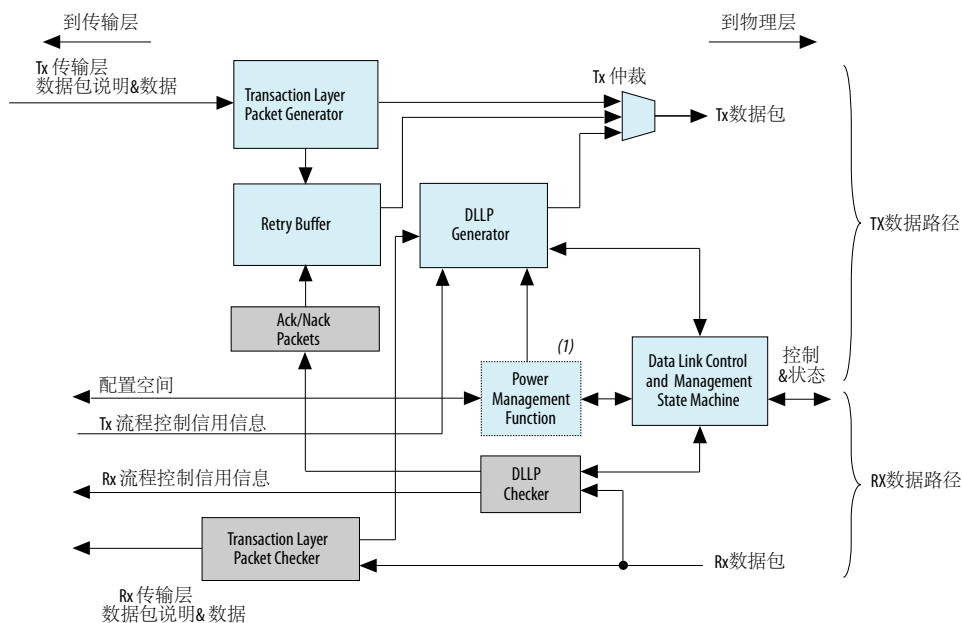
9.2. 数据链路层

数据链路层位于事务层和物理层之间。它维护数据包的完整性和在 PCI Express 链路级进行通信（通过 DLL 数据包的发送）。

DLL 实现以下功能：

- 通过接收和发送 DLL 数据包（DLLP）进行链路管理，并用于下列功能：
 - - DLLP 接收和发送的电源管理
 - - 发送和接收 ACK/NAK 包
 - - 通过生成和检查 TLP 和 DLLP 的 CRC 来维护数据完整性
 - - 使用重试（重播）缓冲器的 TLP 重发以防止 NAKDLLP 接收或重播超时。
 - - 重试缓冲器的管理
 - - 链路重培训请求以防因物理层链路训练和状态机（LTSSM）出现的错误。

图 49. 数据链路层



注释:
(1) 不支持L0s (待机) 或 L1 (低功耗待机)。

DLL 具有以下子块 (sub-block) :

- 数据链路控制和管理状态机—该状态机与物理层 LTSSM 状态机和事务层连接。它初始化链路和流程管理信用并向事务层报告状态。
- 电源管理—该功能处理握手 (handshake) 以进入低电源模式。例如基于配置空间寄存器值和已接收电源管理 (PM) DLLP 的过度。所有 Arria 10 Hard IP for PCIe IP 内核 variant 都不支持低电源模式。
- 数据链路层数据包生成器和检查器—此块与 DLLP 的 16-bit CRC 相对应并维护已发送数据包的完整性。
- 事务层数据包生成器—此块生成发送数据包，包括序列号和 32-bit Link CRC (LCRC)。该数据包也被发送到重试缓冲器用于内部储存。重试模式中，TLP 生成器接收来自于重试缓冲器的数据包并生成发送数据包的 CRC。
- 重试缓冲器—重试缓冲器储存 TLP 和重发所有的未确认数据包，以防止 NAK DLLP 接收。重试缓冲器丢弃所有的已应答数据包，以防止 ACK DLLP 接收。

- ACK/NAK 数据包—ACK/NAK 块处理 ACK/NAK DLLP 和生成已发数据包的序列号。
- 事务层数据包检查器—此块检查已接收 TLP 的完整性并生成 ACK/NAK DLLP 传输请求。
- TX 仲裁 — 此块仲裁交易，并按一下优先顺序：
 - 初始化 FC 数据链路层数据包
 - ACK/NAK DLLP (高优先级)
 - 更新 FC DLLP (高优先级)
 - PM DLLP
 - 重试缓冲 TLP
 - TLP
 - 更新 FC DLLP (低优先级)
 - ACK/NAK FC DLLP (低优先级)

9.3. 物理层

物理层是 PCI Express 协议栈的最底层。也是最接近串行链路的分层。它编码并跨链路发送数据包，还接受和解码收到的数据包。物理层利用运行在 2.5 Gbps 的高速 SERDES 接口连接到链路用于 Gen1 实现，2.5 或 5.0 Gbps 用于 Gen2 实现，以及 2.5, 5.0 或 8.0 Gbps 用于 Gen3 实现。

物理层负责以下操作：

- 训练链路
- 加扰/去扰和 8B/10B 编码/解码用于 2.5 Gbps (Gen1)，5.0 Gbps (Gen2)，或者 8.0 Gbps (Gen3) 每 lane 的 128b/130b 编码和解码
- 串化和解串数据
- 均衡 (Gen3)
- 操作 PIPE 3.0 接口
- 实现自动速率协商 (Gen2 和 Gen3)
- 发送和解码训练序列
- 提供硬件自主速率控制
- 实现 lane 自动倒转

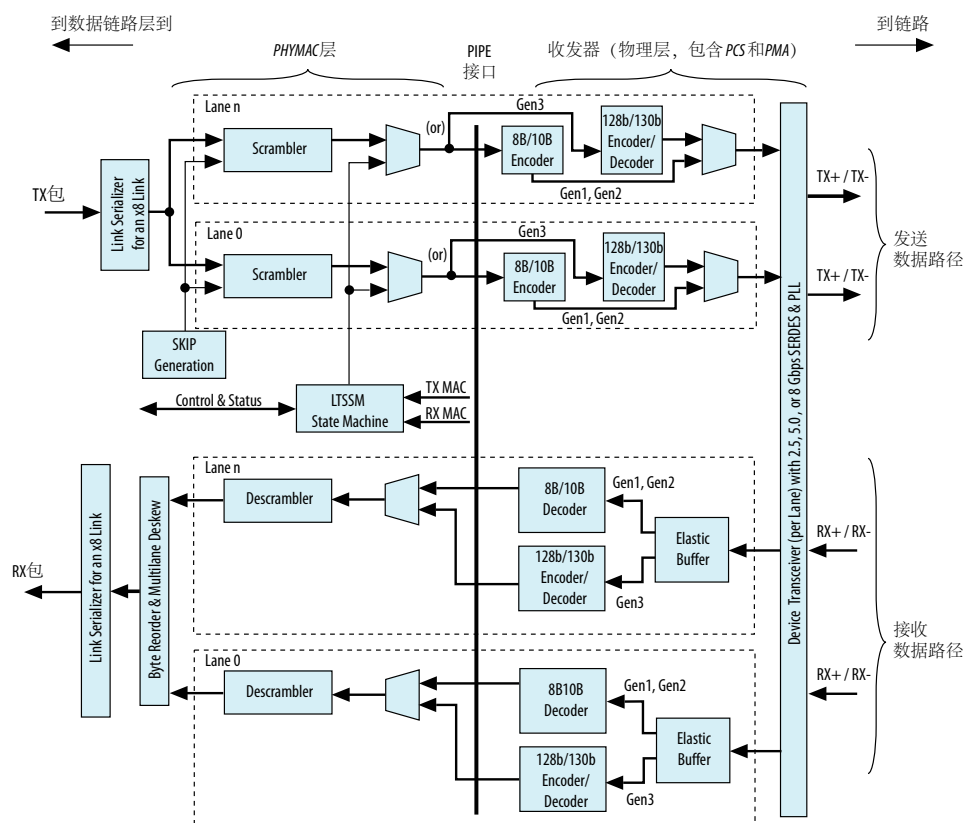
物理层被 PIPE 接口规范细分为两个分层（上图中水平加括）：

- PHYMAC—MAC 层包括 LTSSM 和加扰/解扰字节重排序，和多通道去偏差功能。
- PHY 层—PHY 层包含用于 Gen1 和 Gen2 的 8B/10B 编码和解码功能。还包含用于 Gen3 的 128b/130b 编码和解码功能。PHY 也包含弹性缓冲和串化/解串 (serialization/deserialization) 功能。

物理层集成了数字和模拟两种元件。Intel 设计的 PIPE 接口把 MAC 从 PHY 中分离出来。Arria 10 Hard IP for PCI Express 符合 PIPE 接口规范。

注意：内部 PIPE 接口用于仿真时为可见。但不可用于调试使用逻辑分析器，（如，SignalTap II）的硬件。如果您尝试把 SignalTap II 连接到该接口，将导致无法编译您的设计。

图 50. 物理层体系结构



PHYMAC 块由 4 个主要子块组成：

- MAC Lane—MAC Lane—RX 和 TX 路径都使用此块。
 - 在 RX 侧，此块解码物理层数据包，并向 LTSSM 报告所收到 TS1/TS2 有序集的类型和数量。
 - 在 TX 侧，此块复用来自 DLL 的数据和有序集以及 SKP 子块 (LTSTX)。它还添加 lane 的详细信息，包括初始化期间 LTSSM 禁用 lane 时的 lane 数和 force PAD 值。
- LTSSM—此块实现 LTSSM 以及追踪 TX 和 RX 在各条 lane 上训练序列的逻辑。
- 关于传输，它通过置位全局和每通道控制位与每个 MAC lane 子块和 LTSTX 子块互动从而生成指定物理层数据包。
 - 在接收路径上，它接收由每个 MAC lane 子块报告的物理层数据包。它还使能多通道去偏差块。该块向更高分层报告物理层的状态。
 - LTSTX (有序集和 SKP 生成)—该子集生成物理层数据包。它接收来自 LTSSM 块的控制信号并为每条 lane 生成物理层数据包。它为所有 lane 和用于 TS1/TS2 域中相应链路或 lane 数的 PAD 符生成相同的物理层数据包。该块还通过置位预定义 PIPE 信号及等待其结果来处理 PCS 子层的接收器检测操作。它也在每个预定义时间段生成 SKP 有序集并与 TX 对齐块互动以防止在数据包中间插入 SKP 有序集。
 - Deskew (去偏差)—该子块执行多通道去偏差功能以及已初始化 lane 与数据路径间的 RX 对齐。多通道去偏差实现一个 8-word FIFO 缓冲器用于各 lane 储存符号。每个符号包括 8 个数据位，1 个差异位，和一个控制位。FIFO 丢弃 FTS, COM, 和 SKP 符，并以 D0.0 数据取代 PAD 和 IDL。当全部 8 个 FIFO 都包含数据，就出现一次读取。当多通道去偏差被首先使能时，在检测到第一个 COM 后每个 FIFO 就开始写。如果 7 个时钟周期后所有 lane 都没有检测到 COM 符，lane 就被复位再同步程序重新启动，否则 RX 对齐功能再创建 64-bit 数据发送至 DLL。

9.4. Arria10 Avalon-MM DMA 用于 PCI Express

The Arria10 Avalon-MM DMA for PCI Express IP Core 包含高效 Read DMA, Write DMA, 和 DMA 描述符模块。通常使用 256-bit 接口道应用层的硬件系统的吞吐量是 6 GB/sec 或更高。而使用 128-bit 接口到应用层的硬件系统其吞吐量成相应比列达 3 Gb/sec。

使用 64-byte 负载，但理论上最大吞吐量远少于该量，因为 TLP 头占据了带宽增加的部分。用于 back-to-back TX 存储器写完成的吞吐量，RX 读完成，以及仿真的读和写分别为 2 GB/sec。

注意: 64-byte 数据包是用于以太网的最小数据包。

相关链接

- [Avalon-MM DMA 入门](#) (第 13 页)
- [DMA 描述符控制器寄存器](#) (第 70 页)

9.4.1. 了解内部 DMA 描述符控制器

当在参数编辑器中选择 **Instantiate internal descriptor controller** 时，具有 DMA 的 Avalon-MM 包含一个内部 DMA 描述符控制器以管理读和写 DMA 操作。DMA 描述符控制器包括读和写数据移动器以进行本地存储器读和写。它支持多达 128 个描述符用于读和写 DMA。主机软件利用 PCI Express 主存储器描述符列表中的位置和大小来编程 DMA 描述符控制器内部寄存器。描述符控制逻辑指示 DMA 读逻辑将整个列表复制到其本地 FIFO。

图 51. 使用内部 DMA 描述符控制器的 Qsys 实例设计

该 Qsys 实例设计，**ep_g3x8_avmm256_integrated.qsys** 位于 `<install_dir>/ip/altera/altera_pcie/altera_pcie_a10_ed/example_design/a10` 目录。请参阅 *Getting Started with the Arria10 Avalon-MM DMA* 了解关于仿真和编译该实例设计的说明。出于清晰度，此截屏筛选出了一些接口类型。

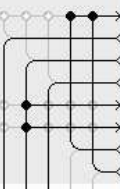


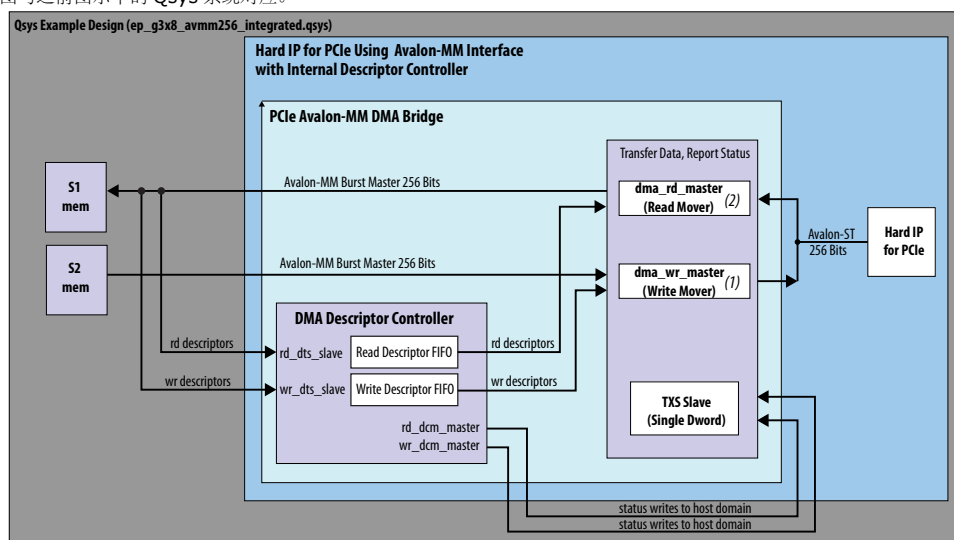
System Contents					Address Map		Interconnect Requirements	
System: ep_g3x8_avmm256_integrated					Path: altpcie_devkit_0			
	Use	Connections	Name	Description	Base			
<input checked="" type="checkbox"/>			DUT	Arria 10 Hard IP for PCI Express				
		txs	Avalon Memory Mapped Slave	0x0000_0000_0000_0000				
		rxm_bar4	Avalon Memory Mapped Master					
		dma_rd_master	Avalon Memory Mapped Master					
		dma_wr_master	Avalon Memory Mapped Master					
		rd_dts_slave	Avalon Memory Mapped Slave	0x0000_0000_0100_0000				
		wr_dts_slave	Avalon Memory Mapped Slave	0x0000_0000_0100_2000				
		rd_dcm_master	Avalon Memory Mapped Master					
		wr_dcm_master	Avalon Memory Mapped Master					
<input checked="" type="checkbox"/>			onchip_memory2_0	On-Chip Memory (RAM or ROM)				
	s1	Avalon Memory Mapped Slave	0x0000_0000_0000_0000					
	s2	Avalon Memory Mapped Slave	0x0000_0000_0000_0000					

图 52. 带有内部 DMA 描述符控制器的 Avalon-MM DMA 框图

该框图与之前图示中的 Qsys 系统对应。



注释:

(1) Write Mover把数据从局部域转移到主机域。

(2) Read Mover把数据从主机域转移到局部域。

本设计使用 BAR0 和 BAR1 创建 64-bit 地址以访问 DMA 描述符控制器。这些 BAR 不能连接任何其他接口。如果 BAR0 必须访问其他接口，就一定要使用一个外部 DMA 描述符控制器。Intel 建议选择内部 DMA 描述符控制器，如果您不打算修改该组件。

用于 64-bit 地址的高性能 BAR2 或 BAR2 和 BAR3 可用于为其他高性能功能接收数据。

相关链接

[Avalon-MM DMA 入门 \(第 13 页\)](#)

9.4.2. 了解外部 DMA 描述符控制器

使用外部 DMA 描述符控制器提供更多灵活性。可以修改或者将它替代以满足您系统的需要。或许您需要修改 DMA 描述符控制器的原因大致如下：

- 实现多通道操作
- 根据链接的列表实现描述符或者实现自定义 DMA 编程模型
- 把描述符储存在本地存储器中，代替系统（主机方面）存储器。

要连接本 variant 中包含的 DMA 逻辑的接口，自定义 DMA 描述符控制器必须实现下列功能：

- 必须与写 Mover 和读 Mover 通信以复制描述符列表到局部存储器。
- 写 Mover 和读 Mover 必须执行储存在本地存储器中的描述符。
- DMA Avalon-MM 写 (WrDCM_Master) 和读 (RdDCM_Master) master 必须能够把状态更新到 TX slave (TXS) 。

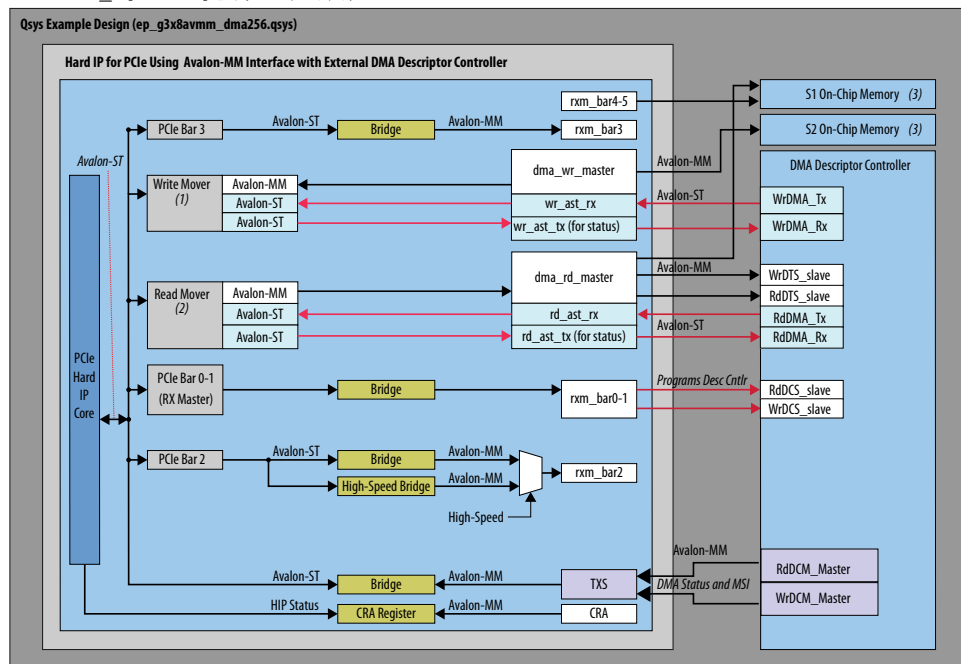
图 53. 带有外部 DMA 描述符控制器的 Avalon-MM DMA 框图

本 Qsys 实例设计，**ep_g3x8_avmm256.qsys**，位于<install_dir>/ ip/altera/altera_pcie/ altera_pcie_a10_ed/example_design/a10 目录中。出于清晰度，此截屏筛选出了一些接口类型。

System Contents Address Map Interconnect Requirements					
System: ep_g3x8_avmm256 Path: DUT					
Use	Connections	Name	Description	Base	
<input checked="" type="checkbox"/>		dma_control_0	Example : DMA Descriptor Cont...		
		RdDCS_slave	Avalon Memory Mapped Slave	0x0000_0000_0000_0000	
		RdDTS_slave	Avalon Memory Mapped Slave	0x0000_0000_0100_0000	
		RdDCM_Master	Avalon Memory Mapped Master		
		WrDCS_slave	Avalon Memory Mapped Slave	0x0000_0000_0000_0100	
		WrDTS_slave	Avalon Memory Mapped Slave	0x0000_0000_0100_2000	
		WrDCM_Master	Avalon Memory Mapped Master		
<input checked="" type="checkbox"/>		DUT	Arria 10 Hard IP for PCI Express		
		txs	Avalon Memory Mapped Slave	0x0000_0000_0000_0000	
		cra	Avalon Memory Mapped Slave		
		rxm_bar0	Avalon Memory Mapped Master		
		rxm_bar4	Avalon Memory Mapped Master		
		dma_rd_master	Avalon Memory Mapped Master		
		dma_wr_master	Avalon Memory Mapped Master		
<input checked="" type="checkbox"/>		onchip_memory2_0	On-Chip Memory (RAM or ROM)		
		s1	Avalon Memory Mapped Slave	0x0000_0000_0000_0000	
		s2	Avalon Memory Mapped Slave	0x0000_0000_0000_0000	

帶有外部 DMA 描述符控制器的 Avalon-MM DMA 框图

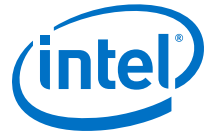
本框图与之前图中的 Qsys 系统对应。当 DMA 描述符控制器被例化为外部组件时，就会在 RdDmaRxData_i[159:0]和 WrDmaRxData_i[159:0]总线上驱动列表条目。



- (1) Write Mover和dma_wr_master把数据从局部域转移到主机域。
- (2) Read Mover和dma_rd_master把数据从主机域转移到局部域。
- (3) 本示例使用片上存储器。然而，此处并不需要片上存储器。

本框图中的 DMA 模块实现以下功能性:

- 读 DMA (Read Mover and dma_rd_master) - 将数据从本地域转移到本地域。利用其高性能主端口发送存储器读 TLP 上游和写完成数据到外部 Avalon-MM 组件。它遵守 *PCI Express Base Specification* 规则中关于标签的考虑, 流程控制信用, 读完成边界, 最大读取量, 以及 4 KB 边界。
- 写 DMA (Write Mover and dma_wr_master) - 将数据从本地域转移到主机域。利用其高性能主端口从 Avalon-MM 从组件读取数据。它通过 Memory Write TLP 发送数据上游。它遵守 *PCI Express Base Specification* 规则中关于标签的考虑, 流程控制信用, RX 缓冲完成规则, 最大负载量, 以及各种 4 KB 边界。
- DMA 描述符控制器 - 管理读和写 DMA 操作。主机软件利用 PCI Express 系统存储器中描述符列表中的位置和大小来编程内部寄存器。描述符控制逻辑指示 DMA 读逻辑将整个列表复制到本地 FIFO。然后每次从 FIFO 取回一个列表条目。它指示正确的 DMA 在本地和主机域之间转移数据。它还通过 TX 从属单 dword 端口 (TXS) 发送 DMA 状态上游。更多关于 DMA 描述符控制器寄存器的信息, 请参考 [DMA 描述符控制器寄存器](#) (第 70 页)。
- RX Master (PCIe BAR0-1) - 允许主机编程 DMA 描述符控制器的内部寄存器。
- TX Slave (TXS) - DMA 描述符控制器向 Avalon-MM 从端口报告每个读和写描述符上的状态。它还使用此端口发送 MSI 请求。



10. 设计实现

完成设计包括额外的步骤来指定模拟性能，管脚分配，和时序收敛。

10.1. 进行管脚约束以分配 I/O 标准到串行数据管脚

在运行 Quartus Prime 编译前，使用 **Pin Planner** 分配 I/O 标准到器件的管脚。

1. 在 Quartus Prime **Assignments** 菜单上，选择 **Pin Planner**。
Pin Planner 出现。
2. 在 **Node Name** 列，找到 PCIe 串行数据管理。
3. 在 **I/O Standard** 列，双击对话框右手角落以调出可用的 I/O 标准列表。
4. 从下列表格中选择合适的标准。

表 76. I/O 标准用于 HSSI 管脚

管脚类型	I/O 标准
HSSI REFCLK	Current Mode Logic (CML), HCSL
HSSI RX	Current Mode Logic (CML)
HSSI TX	High Speed Differential I/O

Quartus Prime 软件将实例约束添加到您的 Quartus Prime 设置文件 (*.qsf)。约束形式为 `set_instance_assignment -name IO_STANDARD <"IO_STANDARD_NAME"> -to <signal_name>。*.qsf 位于您的综合目录中。`

相关链接

[Arria 10 GX、GT 和 SX 器件系列管脚连接指南](#)

了解更多关于 PCB 上管脚连接以及所要求的电阻器值和电压的信息。

10.2. 建议的复位序列以避免链路训练问题

成功的链路训练仅出现在配置 FPGA 后。使用 CvP 的设计进行配置首先加载 I/O 环和外设映像。Arria 10 器件的 Nios II 硬校准 IP 内核可自动校准收发器以在 CvP 完成后及进入用户模式之前优化信号质量。链路训练出现在校准后。请参阅 *Reset Sequence for Hard IP for PCI Express IP Core and Application Layer*（复位用于 Hard IP for PCI Express IP 内核和应用层序列）了解关于控制复位，控制动态重配置，以及链路训练的关键信号说明。Intel 建议用于端点和根端口的复位信号分开控制。成功的复位序列包括下列步骤：

1. 等待配置 FPGA 直到 FPGA 块控制器中显示置位 CONFIG_DONE。
2. 等待 1 ms，在置位 CONFIG_DONE 后，然后置低端点复位。
3. 等待大约 100 ms，然后置低根端口复位。
4. 置低复位输出到应用层。

Intel Corporation. All rights reserved. Agilix, Altera, Arria, Cyclone, Enpirion, Intel, the Intel logo, MAX, Nios, Quartus and Stratix words and logos are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

*Other names and brands may be claimed as the property of others.

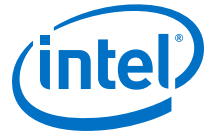
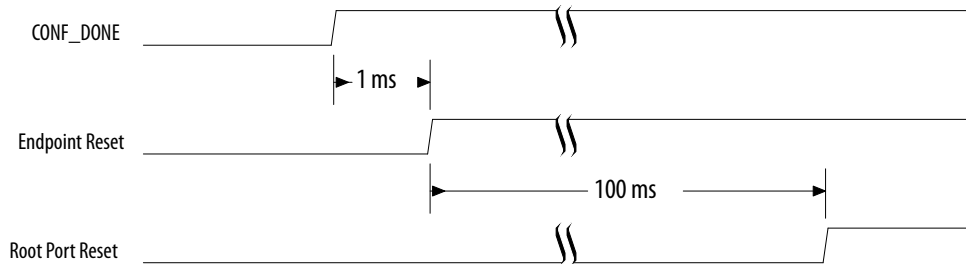


图 55. 建议的复位序列



相关链接

[Intel FPGA Arria 10 收发器 PHY IP Core 用户指南](#)

更多关于自动校准期间 CLKUSR 管脚使用要求的信息。

10.3. 创建 SignalTap II 调试文件以匹配设计层次

对于 Arria 10 器件，Quartus Prime Standard Edition 软件生成两个文件，`build_stp.tcl` 和 `<ip_core_name>.xml`。使用这些文件生成的 SignalTap II 文件具有与设计层次相匹配的探测点。

这些文件储存在 Quartus Prime 软件的 `IP core Directory/synth/debug/stp/` 目录中。

使用 Quartus Prime 软件综合您的设计。

1. 打开 Tcl 控制台，点击 **View > Utility Windows > Tcl Console**。
2. 在 Tcl 控制台中输入以下命令：
`source <IP core directory>/synth/debug/stp/build_stp.tcl`
3. 要生成 STP 文件，输入以下命令：
`main -stp_file <output stp file name>.stp -xml_file<input xml_file name>.xml-mode build`
4. 要将此 SignalTap II 文件（.stp）添加到您的工程，选择 **Project > Add/Remove Files in Project**。然后编译您的设计。
5. 要编程 FPGA，点击 **Tools > Programmer**。
6. 要开启 SignalTap II 逻辑分析器，点击 **Quartus Prime > Tools > SignalTap II Logic Analyzer**。

软件生成脚本可能在 `<output stp file name>.stp` 中没有分配 SignalTap II 采样时钟。因此，Quartus Prime 软件自动创建一个称为 `auto_stp_external_clock` 的时钟管脚。您需要手动将正确时钟信号替换为每个 STP 实例的 SignalTap II 采样时钟。

7. 重编译您的设计
8. 要观察您 IP 核的状态，点击 **Run Analysis**。

可看到着红色的信号或 SignalTap II 实例，表示不可用于您的设计。大多数情况下，可忽略这些信号和实例。由于软件生成了较宽的总线而您的设计中并不包含这些实例。从而出现这类消息。

10.4. SDC 时序约束

您的顶层 Synopsys Design Constraints 文件 (.sdc) 必须包含用于 Arria 10 Hard IP for PCIe IP 内核的时序约束宏。

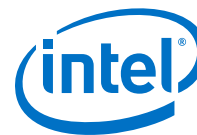
实例-1: 需用于 **Arria 10 Hard IP for PCIe** 及实例设计的 **SDC** 时序约束

```
# Constraints required for the Arria 10 Hard IP for PCI Express
# derive_pll_clock is used to calculate all clock derived
# from PCIe refclk. It must be applied once across all
# of the SDC files used in a project
derive_pll_clocks -create_base_clocks
```

此约束只能位于项目中一个跨所有 SDC 文件的地方。如果这些约束被多次应用则 Fitter 时序分析与 TimeQuest 时序分析间会出现差异。

相关链接

要如何约束针对 Arria 10 ES2, ES3 或生产器件的 PCIe Gen1, Gen2 或 Gen3 设计?



A. 传输层数据包 (TLP) 头格式

下图显示了无数据负载的 TLP 头格式。

图 56. 存储器读请求, 32-Bit 寻址

Memory Read Request, 32-Bit Addressing																																
	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	0	0	0	0	0	0	0	0	TC		0	0	0	0	0	TD	EP	Attr		0	0	Length									
Byte 4	Requester ID																Tag								Last BE				First BE			
Byte 8	Address[31:2]																												0		0	
Byte 12	Reserved																															

图 57. 存储器读请求, 锁定 32-Bit 寻址

	+0								+1								+2								+3												
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0					
Byte 0	0	0	0	0	0	0	0	1	0	TC		0	0	0	0	0	TD	EP	Attr	0	0	Length															
Byte 4	Requester ID																Tag								Last BE				First BE								
Byte 8	Address[31:2]																															0		0			
Byte 12	Reserved																																				

图 58. 存储器读请求, 64-Bit 寻址

	+0								+1								+2								+3											
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0				
Byte 0	0	0	0	0	0	0	0	0	TC		0	0	0	0	TD	EP	Att r		0	0	Length															
Byte 4	Requester ID																Tag								Last BE				First BE							
Byte 8	Address[63:32]																																			
Byte 12	Address[31:2]																																0	0		

图 59. 存储器读请求, 锁定 64-Bit 寻址

	+0								+1								+2								+3									
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0		
Byte 0	0	0	1	0	0	0	0	1	0	TC			0	0	0	0	T	EP	Att r		0	0	Length											
Byte 4	Requester ID																Tag						Last BE				First BE							
Byte 8	Address[63:32]																																	
Byte 12	Address[31:2]																																0	0

图 60. 配置读请求根端口 (Type 1)

Configuration Read Request Root Port (Type 1)																																		
	+0								+1								+2								+3									
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0		
Byte 0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	TD	EP	0	0	0	0	0	0	0	0	0	0	0	0	0	1		
Byte 4	Requester ID																Tag								0 0 0 0				First BE					
Byte 8	Bus Number								Device No				Func				0	0	0	0	Ext Reg				Register No								0 0	
Byte 12	Reserved																																	

图 61. I/O 读请求

I/O Read Request																																
	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	TD	EP	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Byte 4	Requester ID																Tag								0 0 0 0				First BE			
Byte 8	Address[31:2]																											0 0				
Byte 12	Reserved																															

图 62. 不带数据的消息

Message without Data																																
	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	0	1	1	0	r	r	r	0	TC			0	0	0	0	TD	EP	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Byte 4	Requester ID																Tag								Message Code							
Byte 8	Vendor defined or all zeros																															
Byte 12	Vendor defined or all zeros																															

注释:
(1) Avalon-MM 中不支持。

图 63. 不带数据的完成

Completion without Data																																	
	+0								+1								+2								+3								
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	
Byte 0	0	0	0	0	1	0	1	0	0	TC			0	0	0	0	TD	EP	Att r	0	0	Length											
Byte 4	Completer ID																Status		B	Byte Count													
Byte 8	Requester ID																Tag								0	Lower Address							
Byte 12	Reserved																																

图 64. 不带数据的完成锁定

Completion Locked without Data																																		
	+0								+1								+2								+3									
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0		
Byte 0	0	0	0	0	1	0	1	1	0	TC				0	0	0	0	0	TD	EP	Att r		0	0	Length									
Byte 4	Completer ID																Status				B	Byte Count												
Byte 8	Requester ID																Tag								0	Lower Address								
Byte 12	Reserved																																	



A.1. 带有数据负载的 TLP 数据包

图 65. 存储器读请求, 32-Bit 寻址

	+0								+1								+2								+3										
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			
Byte 0	0	1	0	0	0	0	0	0	0	TC		0	0	0	0	0	TD	EP		Att r		0	0	Length											
Byte 4	Requester ID																Tag								Last BE				First BE						
Byte 8	Address[31:2]																															0	0		
Byte 12	Reserved																																		

图 66. 存储器读请求, 64-Bit 寻址

Memory Write Request, 64-Bit Addressing																																	
	+0							+1							+2							+3											
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	
Byte 0	0	1	1	0	0	0	0	0	0	TC		0	0	0	0	0	TD	EP	Att r	0	0	Length											
Byte 4	Requester ID																Tag								Last BE				First BE				
Byte 8	Address[63:32]																																
Byte 12	Address[31:2]																															0	0

图 67. 配置写请求根端口 (Type 1)

Configuration Write Request Root Port (Type 1)																																		
	+0								+1								+2								+3									
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0		
Byte 0	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	TD	EP	0	0	0	0	0	0	0	0	0	0	0	0	0	1		
Byte 4	Requester ID																Tag								0 0 0 0				First BE					
Byte 8	Bus Number								Device No								0	0	0	0	Ext Reg				Register No								0	0
Byte 12	Reserved																																	

图 68. I/O 写请求

I/O Write Request																																	
	+0								+1								+2								+3								
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	
Byte 0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	TD	EP	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
Byte 4	Requester ID																Tag								0 0 0 0				First BE				
Byte 8	Address[31:2]																															0	0
Byte 12	Reserved																																

图 69. 带数据的完成

Completion with Data																																
	+0							+1							+2							+3										
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	1	0	0	1	0	1	0	0	TC			0	0	0	0	TD	EP	Att r	0	0	Length										
Byte 4	Completer ID																Status		B	Byte Count												
Byte 8	Requester ID																Tag								0	Lower Address						
Byte 12	Reserved																															

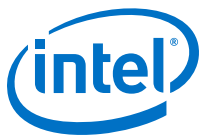


图 70. 带数据的完成锁定

Completion Locked with Data																																								
	+0								+1								+2								+3															
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0								
Byte 0	0	1	0	0	1	0	1	1	0	TC				0	0	0	0	TD	EP	Att r		0	0	Length																
Byte 4	Completer ID																Status				B		Byte Count																	
Byte 8	Requester ID																Tag								0	Lower Address														
Byte 12	Reserved																																							

图 71. 带数据的消息

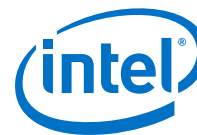
Message with Data																																
	+0							+1							+2							+3										
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	1	1	1	0	r 2	r 1	r 0	0	TC				0	0	0	0	TD	EP	0	0	0	0	Length								
Byte 4	Requester ID																Tag								Message Code							
Byte 8	Vendor defined or all zeros for Slot Power Limit																															
Byte 12	Vendor defined or all zeros for Slots Power Limit																															



B. Arria 10 Avalon-MM DMA 接口 PCIe 解决方案用户指南存档

如有 IP 内核版本未列入，可应用之前版本的 IP 内核用户指南。

IP 内核版本	用户指南
16.0	Arria 10 Avalon-MM DMA Interface for PCIe Solutions User Guide
15.1	Arria 10 Avalon-MM DMA Interface for PCIe Solutions User Guide
15.0	Arria 10 Avalon-MM DMA Interface for PCIe Solutions User Guide
14.1	Arria 10 Avalon-MM DMA Interface for PCIe Solutions User Guide



C. 修订历史

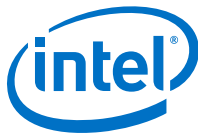
C.1. 具有 DMA 的 Avalon-MM 接口文档修订历史

日期	版本	修订内容
2016 年 10 月	16.1	<p>IP 内核修订内容如下：</p> <ul style="list-style-type: none"> 提高了用于 128-和 256-bit 的最大 DMA 传输到 1 MB。 时序模块最终用于大多数 Arria 10 器件封装。但某些具有扩展温度范围的军事和汽车速度级别除外。 <p>用户指南修订内容如下：</p> <ul style="list-style-type: none"> 将 test_in[31:0] 的建议值从 0xa8 更改为 0x188。 删除了关于连接 pin_perst 的建议。这些建议不适用于 Arria 10 器件。 纠正了全部 <i>Hard IP for PCI Express IP Core</i> 的特性比较列表中所支持的标签数。 增加了 PCIe 分岔 (bifurcation) 到全部 <i>Hard IP for PCI Express IP Core</i> 的特性比较列表中。不支持 PCIe 分岔。 删除了不可用的 DMA 模块 Linux 软件驱动器参考资料。 增加了涵盖设计实例限制的部分。 增加了 -3 到 125 MHz 接口的建议速度等级。
2016 年 5 月	16.0	<p>重新设计的应用层 128-bit 接口为片上和外部存储器持续提供高吞吐量。</p> <p><i>Getting Started with the Avalon-MM DMA Endpoint</i> 章节中，更改了在 Arria 10 GX FPGA 套件-Production (非 ES2) 版本中指定 10AX115S2F4511SG 器件的指令。</p> <p>为 Quartus Prime Pro Edition 软件添加了关于 OpenCore Plus IP 评估的支持。</p> <p>为 Gen3 PIPE 模式添加了使用 ModelSim, VCS, 和 NCSim 仿真器的仿真支持。</p> <p>添加了基础 SignalTap 逻辑分析器文件的自动生成以助于调试。</p> <p>修订了支持 DMA 的 Avalon-MM IP 内核体系结构中关于 DMA 描述符控制器的讨论。</p> <p>修订了 <i>Read DMA Example</i> 以反映 256-bit 接口当前最大传输量为 64 KB。该实例现对应 <install_dir> 中提供的设计实例。</p> <p>更新了 <i>Physical Layout of Hard IP in Arria 10 Devices</i> 中的图示以包含更多关于收发器 bank 和 channel 限制的细节。</p> <p>添加了 Vendor Specific Extended Capability (VSEC) Revision 和 User Device or Board Type ID register from the Vendor Specific Extended Capability 到 GUI 组件 VSEC 选项卡。</p> <p>删除了 <i>Arria 10 PCI Express Quick Start Guide</i> 章节。该章节未提供 DMA 的功能性。</p> <p>更正了关于写描述符 Avalon-MM 从端口的说明。</p> <p>添加了之前版本中缺少的 <i>Vendor Specific Extended Capability (VSEC)</i> 参数说明。</p> <p>添加了关于 Gen3 ES3 器件 bank 组的收发器用途布局限制。</p> <p>删除了 -3 速率级器件的支持。</p> <p>添加了附录罗列本用户指南之前的版本。</p> <p>更正了细微错误及错别字。</p>
2015 年 11 月	15.1	<p>修订内容如下：</p> <ul style="list-style-type: none"> 添加了 256 个选项卡支持以在高延迟设计中提高吞吐量。 添加了 RX 完成缓冲器上溢监控支持。 为提高性能且减少内部缓冲器的要求，限制了描述符大小为 8 KB。 重新设计了 GUI 组件。
继续..		



日期	版本	修订内容
		<ul style="list-style-type: none"> • 添加了新的设计实例选项卡，可用来生成设计实例并下载到 Altera Arria 10 GX FPGA 开发套件。 • 删除了 RX 缓冲器分配参数中的参数值 High 和 Maximum。Avalon-MM 接口不支持这些值。 • 完善了 npwr 的定义。 • 更正了资源利用率。 • 阐明了更改 RC Read Status and Descriptor Base (Low) 和 RC Write Status and Descriptor Base (Low) 寄存器地址前的必要条件。 • 为单个 dword 写添加了立即写模式。此数据储存在 WR_RC_LOW_SRC_ADDR 寄存器中。DMA 描述符新的 Immediate Write Mode 位控制该功能。 • 更正了 <i>TLP Support Comparison for all Hard IP for PCI Express IP Cores</i> 条目。仅支持带数据/不带数据的完成用于 Avalon-MM DMA 接口。不支持带请求/不带请求的消息用于 Avalon-MM 接口。 • 为 Avalon-MM DMA Bridge with Internal Descriptor Controller 和 Avalon-MM DMA Bridge with External Descriptor Controller 图示添加了可选的硬核 IP 状态总线信号。
2015 年 6 月	15.0	在 Physical Layout of Hard IP in Arria 10 Devices 中添加了注释以解释 Arria 10 设计约束的要求：如果用 Gen3 x4 或 Gen3 x8 IP 内核配置器件一侧的下 HIP，且在器件同侧的上 HIP 也用 Gen3 IP 内核配置，那么上 HIP 必须通过 x4 或 x8 IP 内核配置。
2015 年 5 月	15.0	<p>用户指南修订内容如下：</p> <ul style="list-style-type: none"> • 在 <i>Interface System Settings</i> 中添加了 Enable Hard IP Status Bus when using the AVMM interface 参数。IP 内核 15.0 或更新版本中有此参数。
2015 年 5 月	15.0	<ul style="list-style-type: none"> • 添加了 Enable Altera Debug Master Endpoint (ADME) 参数以支持可选 Native PHY 寄存器通过 Altera System Console 进行编程。 • 添加了高达 4 KB 负载的下游突发读请求支持，如果已开启参数编辑器中的 Enable burst capability for RXM BAR2 port。以前的最大下游读请求负载为 512 字节。请参阅 Arria10 Avalon-MM DMA 用于 PCI Express (第 98 页)。 • 在 <i>Device Capabilities</i> 话题中，更正了 Maximum payload size 参数允许用于 Avalon-MM DMA IP 内核 variation 的值。 • 更改所支持的 variation 以支持 Gen3 x2。 • 删除了 RX Buffer credit allocation -performance for received requests 参数的 High 和 Maximum 值。因为已是无效设置。/>。 • 完善了 channel 设置说明，为 Gen1 和 Gen2 数据率添加了 fPLL 设置，还在 Arria 10 器件中 Hard IP 的物理布局 (第 28 页) 中添加主 CGB 位置。 • 恢复了 <i>Design Implementation</i> (设计实现) 章节。 • 在 <i>Features</i> 部分的 Hard IP for PCI Express IP Cores 性能列表中增添了 Avalon-ST Interface with SR-IOV variations 栏。 • 删除了 Migration 和 TLP 格式附录，但新添加了 <i>Frequently Asked Questions</i> (常见问题) 附录。 • 更新了 <i>SDC Timing Constraints</i> 部分的信息。 • 从 设计实例 (第 9 页)。第 1-8 页中删除了静态实例设计列表。可从安装目录中导出可用实例设计的列表 • 修复了细微错误和错别字。
2014 年 12 月	14.1	此 Arria10 用户指南的修订内容如下：

继续...



日期	版本	修订内容
		<ul style="list-style-type: none">在 <i>Getting Started</i> 章节中，更正了仿真的目录路径。添加了 RX Burst Master 仅支持 dword 粒度的实际情况。添加了关于 test_in[2], test_in[6]和 test_in[7]的定义。添加了关于 Quartus II 编译的指令。
2014 年 8 月	14.0 Arria 10	<p>Arria10 Avalon-MM DMA for PCI Express IP 内核的修订内容如下：</p> <ul style="list-style-type: none">修订了描述符控制器的编程模型。添加了将在方针目录中自动生成的仿真日志文件，altpcie_monitor_a10_dlhip_tlp_file_log.log。要在 Quartus II 14.0 中仿真，必须重新生成您的 IP 内核从而创建支持的监控文件，以生成 altpcie_monitor_a10_dlhip_tlp_file_log.log。请参阅 <i>Understanding Simulation Dump File Generation</i> 了解更多信息。添加了支持 128 位和 256 位接口到应用层。添加了 64 位寻址而不再需要地址转换。删除了 <i>Channel Placement for PCIe in Arria 10 Devices</i>。请联络您的 Altera 销售代表了解 PLL 和 channel 使用方法。添加了 BAR2 的可选突发 RX Master 支持。修订了 <i>Read DMA Example</i> 和 <i>Software Program for Simultaneous Read and Write DMA</i> 以便运行于修订后的描述符控制器编程模型。对描述符控制器进行下列优化：<ul style="list-style-type: none">减小负载优化了性能，如 64 字节以太网数据包减少了主机更新的成本开销支持动态主机更新和 DMA 操作同时进行支持选择 Avalon-MM 桥中嵌入式描述符控制器或单独例化添加了通过可选的 Control Register Access (CRA) Avalon-MM 从端口到已选 Configuration Space 寄存器和链路状态寄存器的访问。添加了第三方 BFM 要求用于 Gen3 variant 的阶段 2 和阶段 3 均衡的仿真支持。添加了多个 MSI/MSI-X 支持。 <p>本用户指南修订内容如下：</p> <ul style="list-style-type: none">Arria10 器件中作为可选 refclk 频率的 125 MHz 时钟。Arria10 器件支持 <i>PCI Express Base Specification, Rev 3.0</i> 指定的 100 MHz 参考时钟。更正了 Maximum payload size 参数的值。可用量为 128 或 256 字节。完善了器件 ID 和子系统供应商 ID 的定义以表明这些寄存器仅在 Type 0 (Endpoint)配置空间有效。删除了 Arria10 器件中作为可选 refclk 频率的 125 MHz 时钟。Arria10 器件支持由 <i>PCI Express Base Specification, Rev 3.0</i> 指定的 100 MHz 参考时钟。添加了 <i>Next Steps in Creating a Design for PCI Express</i> 到 <i>Datasheet</i> 章节。删除了 <i>Transaction Layer Protocol Details</i> 章节。此信息仅适用于 Avalon-ST 接口。删除了 PIPE 接口中的 txdatavalid0 信号。此信号不可用。删除了关于 MegaWizard® Plug-In Manager 的参考内容。由于 14.0 the IP Parameter Editor Powered by Qsys 已替代了 MegaWizard Plug-In Manager。添加了关于 test_in[2], test_in[6]和 test_in[7]的定义。更正了数据表： <i>Arria10 Avalon-MM DMA for PCIe</i> 章节里 <i>Performance and Resource Utilization Arria10 Avalon-MM DMA for PCI Express</i> 列表中的接口宽度。删除了关于 pclk 的讨论。Arria 10 器件中客户无法访问该时钟。更正了 <i>Reset and Clocks</i> 章节中 <i>Arria10 器件的 Reset Controller</i> 图示。更正了 CvP Status 寄存器中关于位的定义。删除了 channel 布局图中的 PLL。为 <i>CvP in Arria 10 Devices</i> 图示中支持的配置方案添加了快速被动并行 (FPP)。更新了 <i>Power Supply Voltage Requirements</i> 列表。

继续...

C. 修订历史

UG-01145_avmm_dma | 2016.10.31



日期	版本	修订内容
		<ul style="list-style-type: none"> 纠正了描述符指令总线的名称。字母 <i>DMA</i> 现在更改为 <i>Ast</i>。例如，<code>WrDMARXValid_i</code> 现在为 <code>WrAstRXValid_i</code>。 添加了 <code>RD_CONTROL</code> 和 <code>WR_CONTROL</code> 寄存器 <code>Done</code> 位。设置后，描述符控制器为状态列表中的每个描述符写入该位，并在最后一个描述符完成后发送单个 <code>MSI</code> 突发。 删除了与 Arria10 Avalon-MM DMA 接口 IP 内核无关联的章节。而 Avalon-ST 版本较为完整，可从中找到这些章节： <ul style="list-style-type: none"> – <i>Design Implementation</i>（设计实现） – <i>Optional Features</i>（可选特性） – <i>Debugging</i>（调试） – <i>Throughput Optimization</i>（吞吐量优化）
2013 年 12 月	13.1 Arria 10	首次发布。