

用于AXI4的

用户指南

版本1.09

2023年11月07日

CADENCE机密

© 2023 Cadence Design Systems，Inc.All rights reserved.

Cadence设计系统公司（凯登斯），西利大街2655号美国加利福尼亚州圣何塞市，邮编：95134

**商标**：Cadence Design Systems，Inc.的商标和服务商标本文件中包含的所有信息均以适当的符号归属于Cadence有关Cadence商标的查询，请按上述地址联系公司所有其他商标均为其各自持有人的财产。

**限制许可**：本文档受版权法和国际条约保护，包含Cadence所有的商业机密和专有信息未经授权复制或分发本文档或其任何部分可能会导致民事和刑事处罚。除非本许可声明中另有规定，否则未经Cadence事先书面许可，不得以任何方式复制、复制、修改、发布、上传、发布、传播或分发本文件包含益华或其许可方的专有和机密信息，并根据益华与其客户之间的书面协议提供，且仅可根据该书面协议使用。

除非Cadence另有书面同意，否则本声明授予Cadence客户打印

1. 本文件的硬拷贝，但须符合以下条件：
   1. 本文件不得以任何方式修改
   2. 本文档或其部分的任何授权副本必须包括所有原始版权、商标和其他所有权声明以及本许可声明。
   3. 本文档中包含的信息不得用于开发类似产品或软件，无论是内部使用还是外部使用，也不得用于任何其他方的利益，无论是否考虑。

**免责声明**：本文件中的信息如有变更，恕不另行通知，并不代表CADENCE方面的承诺除非在CADENCE与其客户之间的书面协议中明确规定，否则CADENCE不对本文件所含信息的完整性、准确性或有用性做出任何陈述或保证，并明确否认。CADENCE不保证使用此类信息不会侵犯任何第三方权利，CADENCE不承担任何暗示保证，包括适销性和适用于特定用途。对于因使用此类信息而可能导致的任何类型的损害或费用，CADENCE不承担任何责任。CADENCE客户对客户产品的开发、制造、销售和使用的所有方面拥有全面控制权和最终决策权

**受限权利**：政府使用、复制或披露本网站内容时，须遵守FAR52.227-14和DFAR252.227-7013及以下条款中规定的限制或其继任者。

[关于本文件viii](#_bookmark0)

1. [IP12](#_bookmark2)概述
   1. [概述. 12](#_bookmark3)
   2. [IP12](#_bookmark4)控制器系统概述
   3. [核心配置13](#_bookmark5)
2. [架构概述15](#_bookmark8)
   1. [架构框图15](#_bookmark9)
   2. [控制器PCS16](#_bookmark10)
   3. [PHY逻辑层16](#_bookmark12)
   4. [数据链路层17](#_bookmark13)
   5. [交易层17](#_bookmark14)
      1. [TL接收侧操作18](#_bookmark15)
      2. [TL发送侧操作18](#_bookmark16)
   6. [主机适配层操作18](#_bookmark17)
   7. [应用层18](#_bookmark18)
3. [信号列表和说明20](#_bookmark19)
   1. [引脚列表描述20](#_bookmark20)
4. [时钟和复位21](#_bookmark21)
   1. [IP时钟21](#_bookmark22)
      1. [固定核心时钟（FCC）21](#_bookmark23)
      2. [PM时钟频率22](#_bookmark25)
      3. [RegClock Usage22](#_bookmark26)
   2. [复位信号23](#_bookmark28)
      1. [复位输出23](#_bookmark29)
      2. [link\_down\_handling\_in\_progress23](#_bookmark30)
      3. [热重置输出23](#_bookmark31)
5. [IP寄存器接口25](#_bookmark32)
   1. [概览. 25](#_bookmark33)
   2. [寄存器地址映射26](#_bookmark37)
      1. [IP控制器寄存器](#_bookmark41)
   3. [IP寄存器访问27](#_bookmark42)
      1. [AXI-Lite接口（B路径）27](#_bookmark43)
      2. [APB接口（B路径）28](#_bookmark44)
   4. [通过PCIe链路器29](#_bookmark45)
   5. [配置监听接口29](#_bookmark47)
      1. [CFG写请求流（A路径）30](#_bookmark49)
      2. [CFG读取请求流（A路径）31](#_bookmark50)
6. [寄存器32](#_bookmark51)
   1. [寄存器描述32](#_bookmark52)
7. [控制器和PHY集成解决方案33](#_bookmark53)
   1. [控制器IP集成协议栈（IPS）33](#_bookmark54)
8. [应用接口AXI434](#_bookmark55)
   1. [AXI概述34](#_bookmark56)
   2. [AXI下属34](#_bookmark57)
      1. [AXI从属操作类型34](#_bookmark58)
      2. [AXI从属写入操作35](#_bookmark59)
      3. [AXI从属读取操作38](#_bookmark60)
      4. [未过帐事务处理的标记管理40](#_bookmark61)
      5. [应用层读取大小40](#_bookmark63)
      6. [PF/VF编码41](#_bookmark66)
      7. [AXI从属访问44](#_bookmark70)
      8. [AXI通过区域的44](#_bookmark72)
      9. [AXI通过边带描述符的46](#_bookmark75)
      10. [AXI下属访问说明46](#_bookmark77)
      11. [AXI Subordinate Accessby Examples](#_bookmark79)
      12. [AXI从属（错误）响应52](#_bookmark82)
   3. [AXI经理54](#_bookmark87)
      1. [PCIe到AXI地址转换54](#_bookmark89)
      2. [AXI管理器写访问56](#_bookmark91)
      3. [AXI管理器读取访问60](#_bookmark101)
   4. [AXI链路故障处理66](#_bookmark112)
   5. [原子操作67](#_bookmark113)
      1. [PCIe原子操作67](#_bookmark115)
      2. [AXI-5原子交易](#_bookmark116)
      3. [AtomicOp onAXI Subordinate70](#_bookmark120)
      4. [AtomicOp onAXI Manager71](#_bookmark121)
   6. [信用可用性接口71](#_bookmark122)
      1. [概览. 71](#_bookmark123)
      2. [信用评级72](#_bookmark124)
      3. [信用阈值寄存器72](#_bookmark125)
      4. [链路断开和低功耗模式72](#_bookmark126)
      5. [基于DMA信用的流72](#_bookmark127)
9. [其他接口73](#_bookmark128)
   1. [接口73](#_bookmark129)
      1. [INTx或旧版以太网接口73](#_bookmark130)
      2. [MSI-X中断73](#_bookmark132)
      3. [MSI接口73](#_bookmark134)
   2. [FLR处理74](#_bookmark136)
      1. [HLSFLR处理74](#_bookmark137)
10. [电源管理76](#_bookmark138)
    1. [概览76](#_bookmark139)
    2. [功能电源状态76](#_bookmark140)
       1. [功能电源状态改变握手接口76](#_bookmark141)
    3. [PCIe链路电源状态77](#_bookmark143)
       1. [L0s链路功率状态77](#_bookmark144)
       2. [L1链路电源状态77](#_bookmark145)
       3. [用于L179](#_bookmark147)的寄存器
       4. [L1Substate80](#_bookmark148)
       5. [L1.1操作82](#_bookmark150)
       6. [L1.2操作83](#_bookmark153)
       7. [L1-Substate85](#_bookmark157)的注册详细信息
       8. [L1 Substates87](#_bookmark160)的一般集成详细信息
       9. [L2和L392](#_bookmark164)
11. [综合和物理设计指南94](#_bookmark165)
    1. [合成设置94](#_bookmark166)
    2. [技术特定型号94](#_bookmark167)
    3. [CPCS模块94](#_bookmark168)的时钟复用器
    4. [在物理设计94](#_bookmark169)
    5. [SynthesisApp笔记95](#_bookmark170)
12. [集成指南97](#_bookmark171)
    1. [启动顺序97](#_bookmark172)
    2. [重置序列97](#_bookmark174)
       1. [复位参考框图98](#_bookmark176)
    3. [精密时间测量98](#_bookmark178)
       1. [特征99](#_bookmark179)
       2. [功能说明99](#_bookmark180)
    4. [SRAM集成102](#_bookmark181)
    5. [重要产品数据（VPD）集成103](#_bookmark182)
       1. [VPD读取示例103](#_bookmark184)
       2. [VPD写入示例104](#_bookmark185)
    6. [热插拔集成104](#_bookmark186)
       1. [概览. 104](#_bookmark187)
       2. [信号描述106](#_bookmark190)
       3. [Cadence PCIe控制器热插拔实施108](#_bookmark192)
       4. [寄存器描述110](#_bookmark194)
    7. [IP接口处的PF/VF编码121](#_bookmark195)
       1. [HLSPF/VF编码121](#_bookmark196)
13. [编程指南124](#_bookmark197)
    1. [IP寄存器编程124](#_bookmark198)
    2. [软件驱动程序设置124](#_bookmark199)
    3. [链路均衡设置124](#_bookmark200)
       1. [RP模式下的Gen3链路均衡设置124](#_bookmark201)
       2. [RP模式下的Gen4链路均衡设置125](#_bookmark202)
       3. [RP模式下的Gen5链路均衡设置125](#_bookmark203)
       4. [EP模式下的Gen3链路均衡设置125](#_bookmark204)
       5. [EP模式下的Gen4链路均衡设置125](#_bookmark205)
       6. [EP模式下的Gen5链路均衡设置126](#_bookmark206)
    4. [错误处理126](#_bookmark207)
       1. [接收端错误（REQ）126](#_bookmark208)
       2. [错误128](#_bookmark209)
       3. [HLS错误128](#_bookmark210)
14. [电子指南129](#_bookmark211)
    1. [链接培训129](#_bookmark212)
       1. [链路训练状态寄存器129](#_bookmark213)
       2. [链接错误状态寄存器132](#_bookmark214)
       3. [抗扭斜状态寄存器143](#_bookmark215)
       4. [链路均衡控制设置寄存器144](#_bookmark216)
       5. [链路均衡状态寄存器163](#_bookmark217)
       6. [LTSSM UART控制寄存器170](#_bookmark218)
       7. [LTSSM转换冻结、暂停控制状态寄存器172](#_bookmark219)
       8. [LTSSM转换原因状态寄存器178](#_bookmark220)
       9. [7位LTSSM状态编码180](#_bookmark221)
       10. [L0->恢复LTSSM状态转换原因编码183](#_bookmark222)
       11. [LTSSM转换编码184](#_bookmark223)
       12. [LTSSM状态编码202](#_bookmark224)
15. [核查207](#_bookmark225)
    1. [验证环境207](#_bookmark226)
    2. [验证概述207](#_bookmark227)
16. [附录C：PCIe配置能力链接列表209](#_bookmark228)
    1. [PCIe控制器功能链表实现209](#_bookmark229)
17. [附录G211](#_bookmark231)
    1. [发布版本之间的差异211](#_bookmark232)

针对本指南文件的

本用户指南介绍了将Cadence PCIe控制器IP（HPA）集成到目标设计中的接口和操作细节。本文档假设读者对PCI Express规范有很好的理解。参考[*参考文件和相关文件*](#_bookmark1)。

 重要提示：*“Cadence致力于在我们的代码和通信中使用尊重的语言。我们还积极从现有内容中删除和/或替换不适当的语言。然而，本产品文件可能包含不再被视为适当但仍反映长期存在的行业术语的材料。这些内容将在相关产品/方案术语可以更新而不会对最终用户产生影响时进行处理*

预期受众

本文档旨在帮助参与片上系统（SoC）设计的架构师和采购/工程管理人员

交付包装信息

* 有关信号列表和说明的信息，请参阅交付包的*<package\_path>/docs/cdnhpa\_ips\_IO\_Bullections. html*
* 有关SRAM的信息，请参阅*<package\_path>/docs/cdnhpa\_ips\_sram\_requirements.html*

快递包裹的文件。

* 有关Register Description的信息，请参阅交付包的*<package\_path>/models/ipxact/\**目录
* 有关AXI ID池的信息，请参阅交付包的*<package\_path>/docs/cdnhpa\_ips\_axi\_m\_axid.html*文件

要从linux打开html文件，请键入：

<web\_browser\_name>文件名>.html

参考文献和相关文件

* PCI Express®基本规范修订版1.0、2.0、3.0、4.0。
* PCI Express®基本规范修订版5.0版本1.0。
* PCI Express架构（PIPE）规范的PHY接口（Intel Corporation），版本5.2。
* AMBA® AXI和ACE协议规范版本E。
* AMBA® DTI（*分布式翻译接口*）协议规范版本1.0。
* AMBA® APB方案规范版本C。

缩略语和简称

本文件中使用了以下首字母缩略词和缩写

表1：首字母缩略词

|  |  |
| --- | --- |
| Term | 定义 |
| Aer | 高级错误报告 |
| AES | 高级加密标准 |
| ASPM | 活动状态功率管理 |
| CRC | 循环冗余校验 |
| DLLP | 数据链路层分组 |
| EOP | 数据包结束 |
| FLR | 功能级复位 |

表1：首字母缩略词

|  |  |
| --- | --- |
| Term | 定义 |
| 哈尔 | 主机适配层。应用接口 |
| HLS | HAL流接口（基于CXS） |
| HPA | 高性能架构（第二代Cadence PCIe IP） |
| Ido | 基于ID的订购 |
| LCRC | 链路循环冗余校验 |
| LGA | 传统架构（第一代cadence PCIe IP） |
| LTR | 延迟容差报告 |
| LTSSM | 链路训练和状态状态机 |
| OBFF | 优化的缓冲液冲洗/填充 |
| PCIe | 外围组件接口 |
| PNP | 员额/非员额 |
| RCD | 受限CXL设备模式。以前称为CXL1.1设备模式。局限性包括缺乏热插拔支持以及仅支持68 B微片。枚举为RCiEP。 |
| RCH | 受限CXL主机模式。以前称为CXL1.1主机模式。 |
| SOP | 分组开始 |
| TD | TLP摘要字段 |
| TLP | 事务层分组 |
| TS | 拖车尺寸字段 |
| UR | 不支持的请求 |

获得帮助

如果您在使用本产品或文档时遇到问题，可以向Cadence支持部门提交SR（*服务请求*）这样做时，您必须提供：

* 关于问题的足够信息，以便有效地调查
* 完整描述问题，给出您正在使用的IP版本，并说明问题发生的确切情况

 注意事项：

有关如何使用Cadence产品的查询，请参阅安装在您的网络上的产品文档

服务请求

SR是您提供反馈、提出问题、获得解决方案和报告问题的方式。除非另有说明，否则Cadence支持人员将回复您的服务请求。如果Cadence Support无法回答您的问题，Cadence研发人员将介入。

 重要提示：

确保尽可能准确地指定服务请求的严重级别

有三个严重级别：

* **关键**：如果没有问题的解决方案，您将无法继续
* **重要提示**：您可以继续，但您需要解决问题。
* **次要**：你更喜欢有一个解决方案，但你可以等待它。 注意事项：

注意事项：

您可以请求支持以提高问题的严重性级别因此，除非绝对必要且迫切需要立即解决问题，否则不要使用“**关键”**

使用Cadence在线支持

Cadence鼓励您使用*Cadence在线支持*提交请求。通过Cadence在线支持，您还可以跟踪您的开放服务请求。

要使用*Cadence在线支持*提交服务请求：

1. 如果您没有Cadence在线支持帐户，请转到[http://support.cadence.com](http://support.cadence.com/)，并在***新用户？***，单击

***立即注册***。

在***注册新的Cadence帐户页面***上，请仔细阅读说明并输入您的详细信息进行注册。

1. 登录到Cadence在线支持，然后在***支持主页***上的***我的请求***下，单击***创建案例***。

附加信息

有关Cadence PCIe系列中其他内核的详细信息，请联系Cadence客户支持。有关Cadence及其产品的更多信息，请访问：[http://www.cadence.com](http://www.cadence.com/)。

# 第1章：IP概述

# 第1章：IP概述

* 1. 概述

Cadence PCIe系列由一组可配置的PCIe控制器内核组成，设计用于ASIC设备，为PCIe链路提供行业标准接口。本文档描述了第二代PCIe IP控制器，随后在本文档中也称为HPA*（高性能架构）* IP。此IP可以配置为从单通道到16通道的常规PCIe带宽范围。控制器可以配置为以下模式，可通过带引脚进行选择：

* + - EP*（终点）*
    - RP*（根端口）。*

该控制器符合最新的*PCIe规范*。请参阅[*参考文件和相关文件*](#_bookmark1)一节。

* + - 在PCIe物理层，控制器向SERDES设备提供PIPE兼容接口
    - 在应用程序接口处，控制器提供以下接口选项：

- AXI 4总线接口，可选DMA功能。

客户端界面的宽度取决于配置。

PIPE接口在固定的数据PIPE宽度上操作控制器与以下设备一起工作：

* + - 一个FCC*（固定核心时钟）*，即最大PIPE接口时钟频率

Cadence® PCIe控制器支持各种可配置的功能。有关更多信息，请参阅[*核心配置*](#_bookmark6)

* 1. 控制器IP

下图显示了使用EP和RP PCIe控制器IP实现的系统的基本概述每个图显示：

* + - 用于在EP或RP模式
    - PCIe使用IP作为EP或RP模式。

本文档介绍Cadence® PCIe控制器（包括控制器PCS）。它没有描述PMA及其相关PCS。

PMA/PCS

PMA/PCS

控制PCS

控制PCS

PCIe物理逻辑层

PCIe数据链路层

PCIe事务层

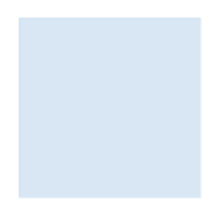
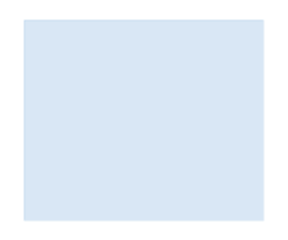
HLS/AX 4

HLS/AX 4

入站

织物

**图1.1：**EP PCIe系统级框图示例



EP PCIe SoC

Cadence PCIe IP

:

:

:

:

x16车道

:

:

:

:

:

:

:

:

x16车道

:

:

:

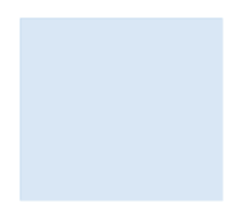
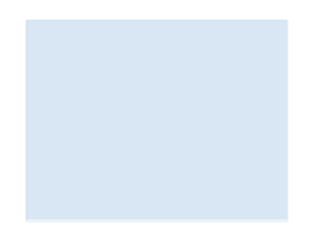
:

Rp

主机处理器

EP

加速器逻辑单元



RP PCIe SoC

Cadence PCIe IP

:

:

:

:

x16车道

:

:

:

:

:

:

:

:

x16车道

:

:

:

:

主机处理器逻辑

**图1.2：**RP PCIe系统级框图示例

EP

加速器设备

PCIe物理逻辑层

PCIe数据链路层

PCIe事务层

入站

出境

## 核心配置

出境

Cadence PCIe IP包含一系列功能，旨在满足不同ASIC架构的需求，从移动和存储等低功耗应用到高性能服务器，以及需要大总线带宽的加速器应用。[*表1.1*](#_bookmark7)列出了允许用于核心配置的体系结构特性。

表1.1：核心配置详细信息

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 部分 | 特征名称 | 描述 | 默认值 | 值范围 |
| Architecture Selection | SOC目标 | *仅支持ASIC*或*支持FPGA的ASIC*的功能。选择*带FPGA支持的ASIC-*  *端口*，IP应包括FPGA  交付. | ASIC | ASIC，ASIC\_WITH\_FPGA\_ SUPPORT |
| 核心协议支持 | 默认情况下，选择*PCIe*、。 | PCIe | PCIe |
| 核心基础规范版本 | 检查此IP支持的PCIe基本规范版本。 | PCIE 5 | PCIe 5 |
| 核心最大速度 | 设置IP的最大链接宽度。 | 32GT/s | 32 GT/s、16 GT/s、8 GT/s、  5GT/s，2.5GT/s |
| 核心管规格版本 | 请检查此IP支持的PIPE规范版本。 | 管道\_4.4.1、管道5.2 | 管道\_4.4.1、管道5.2  低引脚数SerDes架构 |
| 最大链接宽度 | IP的最大链接宽度 | 16 | 4、8、16个 |
| 独立链接数 | 请检查IP支持的独立链接 | 1 | 一、二、四 |
| 物理功能数量 | 计算物理功能的总数。 | 1 | 一、二、四、八、十六 |
| 虚拟功能 | 计算虚拟函数的总数。 | 0 | 2,4,8,16..高达255 |
| PCIe的客户端接口类型 | 确定所需的客户端接口类型。  HLS（HAL流媒体接口），AXI4（AMBA AXI4） | HLS | HLS，AXI4 |
| 客户端（HLS）I/F选择 | 客户端界面宽度 | 根据您选择的PCIe规格版本和链路宽度，从合法选择中恢复 | 512 | 128、256、512 |
| 客户端（AXI 4）I/F选择 | PCIe Manager数据总线宽度 | 设置AXI数据总线宽度。为获得最佳性能，AXI数据总线宽度应等于或  大于内部HLS数据  宽度. | 512 | 128、256、512 |
| AXI PCIe从属数据总线宽度 | 设置AXI数据总线宽度。为获得最佳性能，AXI数据总线宽度应等于或  大于内部HLS数据  宽度. | 512 | 128、256、512 |
| 传递选择 | 管道宽度 | 为此配置选择的管道宽度。 | 32 | 32 |
| RTL模块前缀 | 将您选择的前缀添加到所有rtl模块，以便模块可以是唯一的。 | dbPCIe\_ | dbPCIe\_ |

CRC

硕士

TLP生成

编码器

LTSSM

重试

PMA/PCS

控制器PCS

PCIe PLL

PCIe DLL

PCIe TL

HLS/AX 4

HLS/AXI 4接口

Misc

HLS/AXI 4接口

# 第2章：体系结构概述

第2章：体系结构概述

* 1. 架构框图

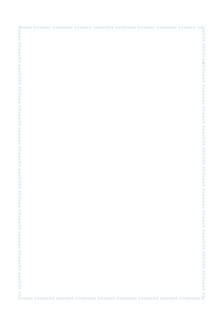
本节介绍Cadence PCIe内核的内部功能块和外部接口的操作控制器IP由以下关键层组成：

* + - CPCS（*控制器PCS*）
    - PLL（*PHY逻辑层*）
    - *数据链路层*
    - TL（*事务层*）
    - *应用层*

PCIe控制器IP HPA架构经过优化，可在接收和发送路径上并行处理多个事务层数据包和多此架构的关键特性是通过减少在IP的不同内部块之间进行的数据重新对齐的量来优化设计中的延迟。

PCIe控制器IP有两个关键接口边界：

* + - 连接PMA/PCS
    - 客户端接口的选择;HLS（HAL流接口）或连接到应用逻辑的AXI 4 I/FPCIe控制器IP对这些接口使用行业标准规范。



Cadence PCIe IP

PCIe控制器

:

:

:

:

x16车道

:

:

:

:

:

:

:

:

x16车道

:

:

:

:

**图2.1：**PCIe控制器IP架构概述

PCIe配置IP寄存器

奴隶

TLP记录

解码器速率匹配FIFO

APB或AXI-Lite

* 1. 控制器PCS

实现控制器PCS IP以支持PCIe PIPE规范5.2（具有低引脚数的SerDes架构）。该逻辑是按通道实现的，并且还支持多链路特性。该模块一方面与符合PIPE 5.2的PHY交互，另一方面使用自定义PIPE 4.4.1接口在内部与控制器IP之间进行数据传输

CPCS以20位（Gen 1/2）和40位（Gen 3/4/5）接口连接到PHY在PIPE 4.4.1上，接口宽度固定为32位。该模块支持标称空RM（速率匹配）FIFO，以减少延迟。它还支持从接收器到发射器的内部可编程选择的测试模式。

[*图2.2*](#_bookmark11)显示了控制器PCS的框图。



RX路径

8b/10 b解码器

Gen1/2 CTC FIFO

逗号对齐器

RX MUX

RX缓冲器

EIEOS校准器

Gen3/4/5 CTC FIFO

TX路径

8b/10 b编码器

TX DEMUX

TX MUX缓冲器

128 b/130b编码器

此块模块支持：

**图2.2：**控制器PCS框图

* + **固定核心时钟模式**：在此操作模式下，控制器PCS需要固定的核心时钟输入，无论PIPE接口上的当前操作速度如何，该输入都提供给核心块这种模式增加了各种时钟传输FIFO，从而增加了设计中的延迟。
  1. PHY逻辑层

物理层的接收操作如下：

1. 数据从CPCS块通过一个或多个通道到达PIPE接口，每个通道通过一个32位接口
2. 来自每个通道的数据被独立地解扰，并通过逻辑来检测链路功率状态转换。
3. 然后，通过将数据传递通过在SKP有序集（Gen 1/2）和SDS有序集（Gen 3及更高版本）上对齐的FIFO来消除泳道的通道作为一个单元对齐。
4. 去偏斜数据由帧解码器解码
5. 帧解码器从数据包中删除SOP/EOP帧定界符，并将它们在内部数据路径上对齐。帧解码器旨在处理不同的链路宽度和通道上所有可能的数据包对齐。
6. 解码后的数据被传递到数据链路层，并带有数据包类型和检测到的错误（如果有的话）等指示
7. 从每个通道接收的数据还被路由到链路训练接收状态机，该链路训练接收状态机负责检测和解码从通道接收的训练序列。
8. 这些状态机中的每一个都将从训练序列中提取的信息传递给LTSSM。

物理层的传输操作如下：

1. 数据来自数据链路层。
2. 数据通过帧编码器，帧编码器将SOP和EOP分隔符添加到数据包并对数据进行条带化

在车道上帧编码器能够处理不同的链路宽度和通道上的数据包的所有合法对齐

1. 来自帧编码器的输出分组与LTSSM生成的训练序列复用
2. 多路复用器在链路训练期间禁用来自帧编码器的数据路径它允许LTSSM控制通道。每个通道都有一个单独的加扰器，在进入PIPE接口之前对其数据进行转换。
3. 预编码块执行PCIe规范中定义的接收和发送预编码对于Gen 1/2/3/4/5，

朝向链路层

控制路径

LTSSM

去偏斜

解扰器

帧解码器

Rx车道维修

Rx预编码器

PIPE Rx FIFO

PIPE RX

从链路层

控制路径

Tx车道恢复

车道扰频器

PIPE Tx

Tx预编码器

PIPE Tx FIFO

帧解码器

**图2.3：**物理层框图

* 1. 数据链路层

DLL的接收操作如下：

1. 来自链路的数据通过每个链路的解码器状态机
2. 解码器通过将接收到的CRC与生成的CRC进行匹配，并将它们的序列号与预期值（对于TLP）进行比较，来验证数据包的完整性
3. 在比较并移除链路层CRC之后，DLLP解码器基于数据包类型将接收到的数据包传递到其
4. 所有接收到的TLP及其序列号和LCRC字段被移除，并被传递到事务层。
5. 数据链路层确认ACK和NAK在数据链路层本身内处理，并且信用DLLP被发送到流控制模块。
6. 在CRC校验之后，接收到的数据链路层应答被发送到数据链路层的发送侧进行处理。
7. 发送端的逻辑将应答与未完成的数据包相匹配，并处理所有错误条件。
8. 接收到的TLP的ACK和NAK被发送到数据链路层的发送侧，在那里它们与传出TLP复用。

DLL的传输操作如下：

1. 通过插入序列号和CRC，对从事务层接收的数据进行格式化，以便传输到物理层
2. 格式化的TLP与诸如ACK/NAK和信用分组的传出DLLP复用，并且通过由核心的两个链路共享的数据路径发送到物理层。发送端还包含与链路相关联的重放缓冲区，其负责在需要时重传数据包。重放缓冲区使用外部单端口RAM来存储数据包。单端口RAM用作重放缓冲器。它既可以写，也可以读。它从不同时写入和读取。还有一个内部指针RAM，用于跟踪存储在重放缓冲区中的数据包。

数据链路层还包含用于生成功率管理DLLP的逻辑，以促进链路到L1和L2状态的转换，作为PCIe功率管理的一部分。

* 1. 事务层

事务层（TL）处理TLP缓冲、排序、信用门控、PCIe标记检查和错误检查。

* + 1. TL接收端操作

以下是在事务层接收端完成的操作的高级细节

1. 对TLP进行解码并检查是否存在畸形错误、长度不匹配、ECRC错误。
2. 存储接收到的TLP的顺序它用于在从接收器缓冲区读取数据包并将其传输到应用程序逻辑时强制排序
3. TLP存储在存储转发RAM中。如果在一个周期中接收到小的TLP，则使用128位数据段来有效地利用RAM有一个单独的接收缓冲RAM，用于Posted、Non-Posted和Completion数据包。
4. TLP被分成以下单独的流并转发以供进一步处理：
   * 过帐.
   * 未发布。
   * 建成
5. 核心的接收流控制参数（用于三种类型的分组的有效载荷和报头信用：发布、非发布和完成）基于接收缓冲器（存储在RAM中）中的可用空间来设置流控制协议确保接收缓冲区不溢出。接收缓冲区将其状态传送给

流控制模块，使得当分组被转发出接收缓冲器时，相应的信用变得可用并且可以被通告给链路。

1. 基于该配置，可以从PCIe链路侧接收多达8个DW对齐的TLP事务层基于所选择的配置将这些分组转换为每个周期128/64位对齐的1/2/4 TLP。
   * 1. TL发送侧操作
2. Posted、Non-Posted和Completion（SC）数据包通过不同的接口从应用程序逻辑到达
3. 事务层中的逻辑对数据包进行多路复用，如果启用，则插入ECRC，并将它们转发到数据链路层。
4. 核心内生成的完成和消息也在同一数据路径上复用到数据链路层。
5. 如果从远程设备没有足够的信用可用，则信用门控逻辑阻止TLP
6. TLP的PCIe排序是从HLS（已发布、未发布、完成）和中断接口向该层中的PCIe链路强制执行的
   1. 主机适配层操作

主机适配层（HAL）中的以下功能简化了应用与PCIe IP的集成

1. 目标PF和VF在比较所有BAR寄存器和TLP地址后解码
2. 存储从出站请求到PCIe链路的PCIe标签，并与完成进行比较，以进行错误检查。如果在编程时间内未从PCIe链路接收到完成，则这些也用于生成完成超时。
3. 端点预期接受来自PCIe链路的所有完成如果接收完成缓冲区的空间不足，则此层中存在的节流逻辑这是一个有用的功能，如果应用逻辑从PCIe链路回压TLP。
4. 对配置寄存器的所有读写请求都被路由到请求所寻址的函数的寄存器集，并将完成包返回到链路。
5. 所有与中断相关的消息都由一个单独的中断处理模块处理，该模块控制内核的UART接口。错误消息由错误处理模块处理还有一些模块可以基于可编程寄存器在内部生成消息。
6. 当收到任何功能的FLR时，此层会自动为所有未完成的请求创建FLR状态为AL的完成
7. 用于针对在IP中检测到的不支持的请求向PCIe链路生成UR完成的逻辑
   1. 应用层

核心的应用层为在IP之外实现的应用逻辑提供接口，使您免受PCI Express协议的复杂性的影响应用层提供以下关键接口：

AXI4接口

AXI4接口通过AXI4桥逻辑连接到现有的HLS接口。该接口基于AMBA AXI4规范。此桥接器实现所有PCIe和AXI4排序规则，任何例外都明确说明。有关更多信息，请参见*“应用接口AXI4”一章*

APB/AXI-Lite

 注意事项：

内容适用于配置期间选择APB还是AXI-Lite

IP通过32位用户可选APB/AXI-Lite总线提供对寄存器实现的PCIe内核的访问。通过该总线，本地处理器可以访问和修改IP的内容，并且访问与PCIe功能相关联的PCIe规范配置的寄存器。核心遵守*AMBA®APB协议规范或AMBA®AXI和ACE协议规范问题E中定义的协议。*

# 第3章：信号列表和描述

第3章：**信号列表和描述**

3.1引脚列表说明

有关信号列表和说明的信息，请参阅交付包的*<package\_path>/docs/cdnhpa\_ips\_IO\_Bullections. html*

要从linux打开html文件，请键入：

<web\_browser\_name>文件名>.html

# 第4章：时钟和复位

第4章：**时钟和复位**

* 1. IP时钟

Cadence IP具有以下两个关键时钟架构。

* + 1. 固定核心时钟（FCC）

在这种模式下，IP以固定频率工作，该频率是最高的PIPE时钟频率，而与PIPE I/F的工作速度无关。这些时钟如[*图4.2*](#_bookmark24)所示

表4.1：时钟域

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 标签 | 域名 | 源 | 同步性 | 频率 |
| 一 | 截止时钟域 | PHY CDR | 与所有时钟 | CDR恢复时钟随PCIe操作速度和PIPE I/F宽度而 |
| B | PIPE I/F时钟域 | PHY参考时钟 | 同步至参考时钟 | 随PCIe操作速度和PIPE I/F宽度而异   * 62.5 Gen1的MHz * 125MHz，适用于Gen2 * 250MHz，适用于Gen3 * 500MHz，用于Gen4 * Gen5为1GHz |
| C | 核心时钟域 | PHY参考时钟 | 同步至参考时钟 | 最大工作PIPE I/F频率 |
| D | AXI I/F时钟域 | 用户 | 与所有时钟 | 高于或等于内核时钟频率 |
| E | 电源管理（PM）时钟域 | 用户 | 与所有时钟 | 最高内核时钟频率。范围：2MHz至60MHz |
| F | APB I/F时钟域 | 用户 | 与所有时钟 | 最高内核时钟频率 |

 注：下图仅用于代表性目的，根据所选配置，块适用。

Cadence PCIe IP

S C P

l r t

C

4

I X A

R r

er e

是的

一个一 个

La L

L l

an

ck o

In i

It

g Lc

O a

L a s t

是啊

h Dr

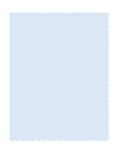
Pe T

咿呀咿呀

I CI

C PC

P P



B

C

D

:

:

:

:

x16车道

:

:

:

:

:

:

:

:

x16车道

:

:

:

:

E

F

一

PMA/PCS

入站

PM I/F APB/I/F



A. 恢复时钟

B.pipe\_pipe\_in C. 核心时钟 D. AXI时钟E.pm\_clock F. apb时钟

下午

APB

出境

**图4.1：**IP时钟

入站

入站

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A.  恢复时钟 | B.  管道安装 | C.堆芯 | D. AXI时钟 | E.公司简介 | F. apb\_bits |

**图4.2：**FCC PCIe IP，带AXI和HLS I/F（PL TX/RX FIFO已分配）

R r

er e

是的

一个一 个

La L

L l

an

ck o

In i

It

g Lc S

o a L

L a s

t H

是啊

h D r

Pe T

咿呀咿呀

IC I

C PC

P P

R r

er e

是的

一个一 个

La L

L l

an

ck o

In i

It

g Lc

O a

L a s t

是啊

h D r

Pe T

咿呀咿呀

IC I

C PC

P P

S C P

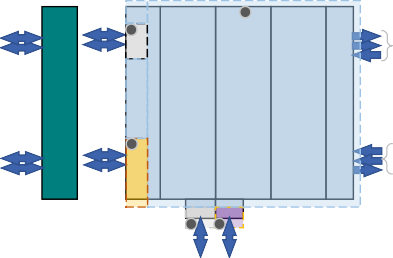
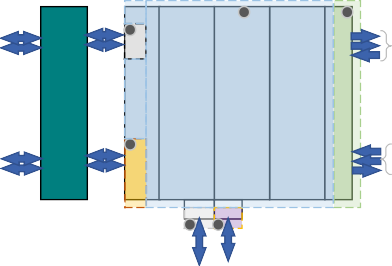
l r t

C

S C P

l r t

C



Cadence PCIe IP

Cadence PCIe IP

C

D

C

一

一

:

:

:

:

x16车道

:

:

:

:

:

:

:

:

x16车道

:

:

:

:

:

:

:

:

x16车道

:

:

:

:

:

:

:

:

x16车道

:

:

:

:

B

B

E F

E F

PM I/ APB/

F I/F

PM I/ APB/

F I/F

PMA/PCS

下午

APB

AXI4

出境

PMA/PCS

下午

APB

出境

* + 1. PM时钟频率

在L1-子状态下，可以关闭*core\_flag*。L1子状态机需要一个单独的时钟（*pm\_CLK*）来运行。在配置器中选择此时钟的默认频率，客户端固件需要对本地管理寄存器中的*pm\_CLK频率寄存器/pm\_CLK频率选择*字段进行编程，以更改*pm\_CLK*频率。仅当LTSSM不在L1中时才支持*pm\_*frequency的更改控制器支持高达2MHz至60MHz的*pm\_sound*频率。重置值反映了在控制器配置期间选择的*pm\_*frequency

 注：*pm\_sms*应定时在62.5MHz，并相应生成控制器SDC文件。 如果定时将以不同的频率关闭，则需要相应地更新SDC。

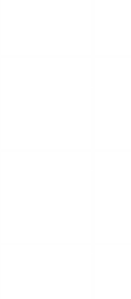
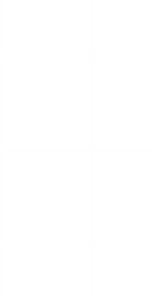
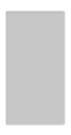
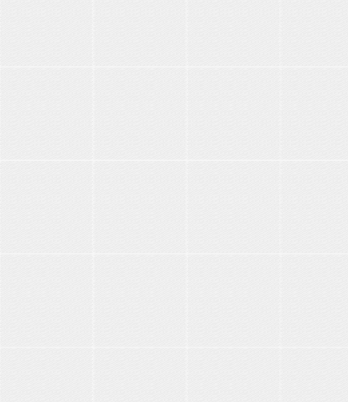
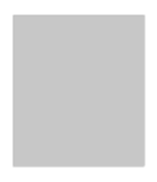
* + 1. 注册时钟使用

控制器有一个单独*的reg\_config输入*，供核心中的PCIe配置空间和ip-mgmt寄存器使用。具有AXI接口的控制器具有reg\_axi\_flag，其由控制AXI层的寄存器使用。寄存器的独立时钟输入允许固件访问寄存器，即使控制器中没有主时钟。当*core\_config*可用时，*reg\_config*应该连接到*core\_config*。当*axi\_axi*可用时，*reg\_axi\_axi*应该连接到*axi\_axi*。

有关时钟端口的详细信息，请参阅软件包中的*IO\_IO. html*

如果您的应用不需要在低功耗状态下访问寄存器，或者低功耗状态始终被禁用，则主时钟可以直接连接到相应的寄存器时钟。

下图显示了在IP外部的客户应用程序逻辑中*reg\_axi*\_req和*reg\_axi\_req*复用的示例



cdnhpa\_ips

公司简介

[force alternate reg\_]

PMA

CPCs

公司简介

0

堆芯

注册\_登录

备用寄存器时钟

1

[axi关闭条件]

轴向

reg\_axi\_axis

1

0

阿西桥

cxl\_top

cdnpcie\_core

[enable alternate reg\_flag] [core\_flag off condition]

CMUX

CMUX

**图4.3：**Reg时钟



时钟多路

CMUX

当链路处于L1\_substate时，*core\_socket*关闭。在这种状态下，*reg\_CLK*可以连接到*备用寄存器时钟*。如果在任何情况下*axi\_axi\_axi都*被选通到IP，则*reg\_axi\_axi\_axi*可以连接到备用时钟。

 注：任何时钟访问寄存器，而主时钟关闭。 频率必须比*core\_bits*或*axi\_bits慢*，具体取决于哪一个更慢。

图中逻辑的描述

[**core\_trigger off condition**]= low\_pwr\_l1\_pm\_substate\_out==3'b101 ||low\_pwr\_l1\_pm\_substate\_out ==3'010

[**force alternate reg\_clk**]：：应用程序逻辑中实现的系统寄存器位，以强制*reg\_clk*的备用时钟。对调试很有用。

[**enable alternate reg\_clock**]：：在应用逻辑中实现的系统寄存器位，用于启用切换到备用时钟。用于调试以防止切换到备用时钟。

[**axi\_trigger off condition**]：：系统寄存器或指示*axi\_trigger*关闭的逻辑

**Alternate reg\_flag**：：主时钟关闭时访问寄存器的任何时钟频率必须低于*core\_error*或

*axi\_axi*，取决于哪个更慢。

* 1. 复位信号

有关PCIe控制器重置信号的信息，请参阅交付包的/doc目录

* + 1. 复位输出

复位信号作为状态信息输出到客户端逻辑客户端逻辑可以使用该信息作为其复位定序器的一有关控制器IP的复位输出信号的信息，请参阅交付包的*/doc*

* + 1. 链路中断处理进行中

当检测到链路断开事件时，*link\_down\_handling\_in\_progress*输出信号在客户端接口时钟上驱动控制器在内部自主执行同步复位。请查看链路中断处理部分，了解应用程序端的链路中断握手。该信号保持有效，直到link\_down握手完成。

* + - * 控制器中的所有内部状态机、定时器和控制寄存器在链路断开复位时复位
      * 在PCIe定义的配置寄存器空间中，所有寄存器（除了**粘性**的那些寄存器之外）都被重置，

*链路断开复位*。

* + - * 本地管理寄存器不会在*链路断开复位*时被清除。这些寄存器保存其编程值，无需重新编程。
      * *link\_down\_handling\_in\_progress*输出被发送到链路断开事件的客户端逻辑。例如，如果链路断开，则客户端逻辑可以重置其自己的FIFO、寄存器或状态机由于此链路断开事件，可能还需要为某些PCIe配置空间寄存器设置

 注意事项：

PHY不需要在*link\_down\_handling\_in\_progress*时重置，尽管这可以用于调试目的。

* + 1. 热复位输出

热复位状态的目的是允许使用带内信令来复位已配置的链路和关联的下游设备热重置允许根端口仅重置PCIe层次结构或设备的选定部分。

当在接收到的训练序列中检测到来自主机的HOT\_CLK时，在端点模式下，在CORE\_CLK上驱动*hot\_reset\_out*输出信号此重置会导致控制器中发生以下事件。

* + - * 发送到客户端逻辑的*hot\_reset\_out*输出指示由于来自主机的热重置指令而发生了链路断开事件
      * 控制器在发生热复位事件时自主执行PCIe配置寄存器的同步复位
      * Client接口上未完成的请求的处理方式与*link\_down\_handling\_in\_progress*类似。
      * 热复位使LTSSM进入检测状态（即Link Down.）。*link\_down\_handling\_in\_progress（链路中断*处理中）

总是与*hot\_reset\_out*一起断言。Host\_reset\_out仅在8个时钟周期内有效

* + - * 客户端可以选择忽略HOT\_OUT输出，仅使用*link\_down\_handling\_in\_progress*

输出.

# 第5章：IP寄存器接口

第5章：**IP寄存器接口**

* 1. 概述

控制器IP寄存器块支持访问[*表5.1*](#_bookmark34)中指定的内核地址空间。下表列出了控制器IP中所有寄存器的高级位置。当AXI桥不存在时，保留对AXI地址空间的访问。控制器IP和IP AXI桥/DMA逻辑寄存器可通过以下两条路径访问

* + - PCIe链路访问（A路径）
      * 如果IP作为EP操作，则远程设备可以通过PCIe CFG事务经由使用VSEC能力的间接寻址来访问所有控制器IP寄存器
      * 可以使用IP端口禁用通过此路径访问IP寄存器
    - AXI-Lite/APB访问（B路径） 注意事项：

根据所选配置，寄存器访问协议可以是APB或AXI-Lite

* + - * 应用程序逻辑可以通过直接寻址通过IP上的AXI-Lite端口访问所有IP寄存器
      * 可选APB端口启用时也可以访问这些寄存器。端口使用AXI-Lite到APB网桥，该网桥内置在IP中用于访问。
      * 请注意，对于IDE配置的控制器，IDE流管理和加密核心寄存器不通过此接口访问，而是通过单独的安全APB接口访问。有关详细信息，请参阅IDE概述部分。

表5.1：寄存器空间

|  |  |  |
| --- | --- | --- |
| 各种寄存器空间 | 通过PCIe链接 | 通过AXI-Lite/APB接口访问 |
| PCIe规格定义的TYPE1 CFG空间 | 是的 | 是的 |
| PCIe规格定义的TYPE0 CFG PF | 是的 | 是的 |
| PCIe规范定义的TYPE0 CFG VF空间 | 是的 | 是的 |
| 用于核心逻辑的 | 是（通过VSEC[1](#_bookmark35)1间接） | 是的 |
| 用于AXI桥/DMA逻辑的IP专用寄存器 | 是（通过VSEC[1](#_bookmark35)间接） | 是的 |
| IDE流管理和加密核心寄存器 | 否（通过安全APB接口访问） | 否（通过安全APB接口访问 |

1参见[*通过PCIe链路访问IP寄存*](#_bookmark46)

e g

d i r

B I X A

e g

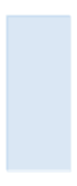
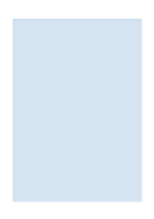
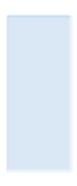
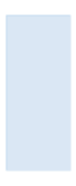
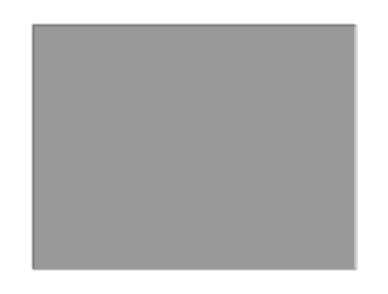
d i r

B I X A

PCIe串行

应用层接口

**图5.1：**IP寄存器接口



A X I B摆脱g e

“n“U n其

其他TLP

RX

HLSRX

TLP

AXI主

M0

RP注册

CFG-

AXILite桥

M1

AXI

LiteInter-connect

CFG CPL TLP

TX

AXITX

B

AXILite-APB

桥

IP寄存器访问接口

配置监听接口

AXI Lite互连

PCIe CFG

一

公司简介

CPCS/PL/DL/TLRX

P C Ie C o re IP

AXI HLS/DTI

RAS寄存器

DMA

寄存器

AXI从寄存器

寄存器

AXIRX

德克萨斯州HLS

CPCS/PL/DL/TLTX

IP核心寄存器

AXI Lite互连

空间

EP PF VF

个reg

IP注册访问接口

在以下部分中，未定义的地址将保留供将来使用，所描述的地址为DWORD地址。在写操作中，各个字节由字节使能位寻址。从链路到保留地址的配置访问导致核心返回具有UR（不支持的请求）完成代码的数据包从本地管理总线到保留地址的读取返回全零，而到保留地址的写入不修改任何寄存器。所有寄存器（保留字段或硬连线字段除外）都可以从本地管理总线写入。

* 1. 寄存器地址映射

控制器解码的地址范围扩展，如下表所示，以支持UCIe规范定义的完整的PCIe系列在下面的其他表格中进一步细分

表5.2：寄存器地址映射

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 访问路径 | 银行解码 | 功能选择 | 寄存器地址- es | IP模式 | 描述 |
| PCIe链路访问 | [25：24][25：24][25：24][25：24] | 2019 - 04 - 23 00：00：00 | 地址[11：0] = 0 - 4095 | EP | PCIe规范配置寄存器类型0 PFn/VFn寄存器 |
| [25：24][25：24][25：24][25：24] | [23：12][23：12][23：12][23：12] | 地址[11：0] = 0 - 4095 | Rp | PCIe规范配置寄存器类型1 |
| [25：24][25：24][25：24] | [23：14][23：14][25：15] | [13：00] = 0 - 16383 | EP | IP核寄存器[1](#_bookmark39) |
| [25：24]=2 [25：24]=3 | X | 地址[17：0] = 0 - 131071 | X | IP AXI/DMA寄存器[1](#_bookmark39) |

表5.2：寄存器地址映射

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 访问路径 | 银行解码 | 功能选择 | 寄存器地址- es | IP模式 | 描述 |
| 应用层APB/AXI-Lite Access2 | [25：24][25：24][25：24][25：24] | 2019 - 04 - 23 00：00：00 | 地址[11：0] = 0 - 4095 | EP | PCIe规范配置寄存器类型0 PFn/VFn寄存器 |
| [25：24][25：24][25：24][25：24] | [23：12][23：12][23：12][23：12] | 地址[11：0] = 0 - 4095 | Rp | PCIe规范配置寄存器类型1 |
| [25：24][25：24][25：24] | [23：14][23：14][25：15] | [13：00] = 0 - 16383 | X | IP核心寄存器，3 |
| [25：24]=2 [25：24]=3 | X | 地址[17：0] = 0 - 131071 | X | IP AXI/DMA寄存器[3](#_bookmark40) |

1. 间接寻址，“addr”取自VSEC Capability Reg0（addr）中编程的值仅当通过IP端口（STRAP\_IPREG\_LINK\_ACCESS\_ENABLE）启用访问时启用
2. 此地址使用BDF格式，符合addr的“功能选择”字段（即[23：12]）中的PCIE规范。
3. 直接寻址，APB/AXI-Lite接口上提供的“addr”

5.2.1 IP控制器寄存器

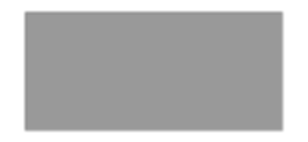
IP控制器寄存器空间根据下表中给出的层和功能进一步分类

表5.3：IP控制器寄存器映射

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [25：24] | [23：14] | [13：10] | REG\_ADDR[9：0] | IP核寄存器空间 |
| 1 | 0 | 0 | 0 - 1023 | 控制器PCS块寄存器 |
| 1 | 0 | 1 | 0 - 1023 | 物理层块寄存器 |
| 1 | 0 | 2 | 0 - 1023 | 数据链路层块寄存器 |
| 1 | 0 | 3 | 0 - 1023 | 事务层块-寄存器 |
| 1 | 0 | 4 | 0 - 1023 | HLS块寄存器 |
| 1 | 0 | 5 | 0 - 1023 | 通用RAS寄存器 |
| 1 | 0 | 6-14 | 0 - 1023 | 储备-未使用 |
| 1 | 0 - 256 | 15 | 0 - 1023 | 每功能核心寄存器 |

* 1. IP注册访问
     1. AXI-Lite接口（B路径）

客户端软件用于访问内核寄存器的控制器IP寄存器接口基于AXI-Lite规范。下图详细说明了接口上的写入和读取时序。



arready

AXI-Lite读取地址

arvalid

信道

araddr

A0

A1

A2

rready

AXI-Lite读取

rvalid

响应通道

Rresp

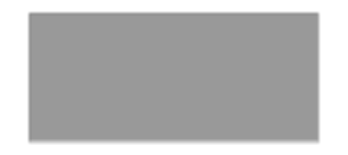
**图5.2：**AXI-Lite接口读取

R2

R1

R0

### APB接口（B路径）



wready

wvalid

AXI-Lite写入数据

信道

wdata

D0

D1

D2

wstrb

S0

S1

S2

bready

AXI-Lite写入响应

巴瓦尔

信道

布雷斯普

R0

R1

R2

**图5.3：**AXI-Lite接口写入

A2

A1

A0

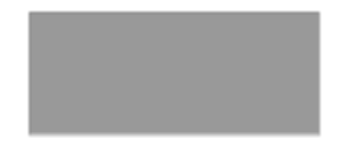
准备好的

AXI-Lite写入地址 awvalid

信道

awaddr

客户端软件访问控制器IP寄存器的控制器IP寄存器接口还支持APB接口，以获得更大的灵活性。如果选择APB，则APB到AXI-Lite的桥接将在核心寄存器块和CDC块中实例化这确保APB写入和读取从APB时钟转换到AXI-Lite时钟域。



PSEL

APB读取控制

可罚的

预写

paddr

A0

普尔达塔

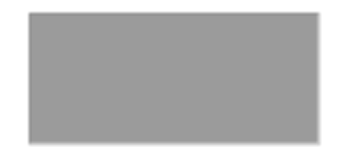
D0

APB读取地址&

数据

普斯尔韦尔

**图5.4：**APB接口读取



PSEL

APB写控制

可罚的

预写

paddr

A0

数据库

D0

APB写入地址，数据

&选通

pstrb

S0

普斯尔韦尔

**图5.5：**APB接口写入

* 1. 通过PCIe链路

如[*表5.2*](#_bookmark38)所示，除了规范强制配置寄存器之外，HPA控制器还包含实现特定寄存器，以帮助内核高效且无错误地运行。这些IP寄存器主要通过APB/AXILite接口访问。HPA控制器还支持通过PCIe链路（主机）访问这些寄存器（包括要启用此功能，必须设置带输入*strap\_ipreg\_link\_access\_enable*。使用配置空间中的供应商特定能力（VSEC）寄存器，使用下面定义的间接机制来实现该功能。

每个这样的访问将包括一个“地址”阶段和一个“数据”阶段。*strap\_ipreg\_link\_access\_enable*strap需要为高才能允许此访问，否则将返回“UR”完成。以下供应商特定寄存器将用于访问：

* + - * 供应商特定地址寄存器（*i\_vendor\_specific\_addr\_reg0*）：用于读取/写入地址
      * 供应商特定数据寄存器（*i\_vendor\_specific\_data\_reg1*）：用于写入数据

1. 对于通过PCIe链路写入IP寄存器，要遵循的步骤是：
   1. 地址阶段：主机发送以“供应商特定地址寄存器0”为目标的配置写入，配置写入有效负载数据应包含目标IP/AXI寄存器的地址。控制器将根据PCIe规则返回用于配置写入的
   2. 数据阶段：主机发送下一个配置写入，目标为“供应商特定数据寄存器1”，配置写入有效负载数据应包含要写入IP/AXI寄存器的实际数据，配置写入TLP中的字节有效也将用于IP寄存器写入访问的控制器将根据PCIe规则返回用于配置写入的“SC”Cpl。
   3. 只有在接收到对“供应商特定数据寄存器1”的第二次配置写入后，控制器才会使用上述步骤中捕获的地址、数据和选通在内部生成IP寄存器写入访问，AxPROT[0]将被设置为0以表示其为配置访问。将更新IP/AXI reg空间中的目标寄存器。
2. 要通过PCIe链路读取IP寄存器，请遵循以下步骤
3. 地址阶段：主机发送以“供应商特定地址寄存器0”为目标的配置写入，配置写入有效负载数据将包含目标IP/AXI寄存器的地址。控制器将根据PCIe规则返回用于配置写入的“SC”
4. 数据阶段：主机接下来发送针对“供应商特定数据寄存器1”的配置读取
5. 只有在接收到对“供应商特定数据寄存器1”的第二次配置读取后，控制器才会使用上述地址阶段捕获的地址在内部生成IP寄存器读取访问

AxPROT[0]将被设置为0以表示它是配置访问。对于读取返回的读取数据和状态将用于根据PCIe规则在步骤2中生成用于配置读取的CplD，以返回IP寄存器读取数据并完成访问。

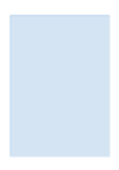
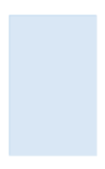
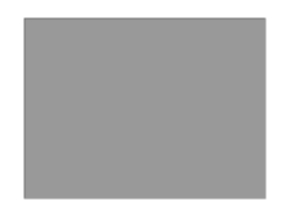
* 1. 配置监听接口

Cadence PCIe控制器按照规范中定义的功能实现PCIe配置空间寄存器。它支持PCI兼容配置空间和PCIe扩展空间寄存器。取决

在配置过程中选择的特征上，控制器自动建立能力结构的链接列表

在枚举过程中，主机可以遍历控制器中的结构链表，以发现支持的功能。根端口软件使用相同的能力结构来获得控制和状态信息。PCIe配置读/写TLP机制用于此目的。控制器将传入的CFG RD/WR TLP映射为对内部实现的配置空间寄存器的读/写访问。能力结构的控制字段被路由到控制器IP的不同层以正确地提供信息。

此配置监听接口的目的是提供灵活性，以便在PCIe配置空间中实现您自己的寄存器集该接口仅侦听从链路接收的传入CFG RD/WR TLP，并将其放置在简单配置侦听读/写接口上。核心使用AXI-Lite实现了一个snoop接口



P C Ie C o re IP

A X I B摆脱g e

“n“U n其

HLS RX

其他TLP

CPCS/PL/DL/TL RX

RX

AXI RX

一

CFG-TLP

**M0 M1**

CFG-

AXILite桥

PCIe串行

A2

PCIe CFG

空间EP PF VF

AA1 Xi

Lite

互连

注册RP注册

AXI

IP核心寄存器

AXI

Lite Inter-conn ect

Lite Inter-conn ect

AXI主寄存器

AXI从寄存器

e g

d i r

B I X A

e g

d i r

B I X A

应用层接口

DMA

寄存器

CFG CPL TLP

TX AXI

HLS TX

CPCS/PL/DL/TL TX

TX

全境通告

AXILite桥

IP寄存器访问接口

A2

IP注册访问接口

配置监听接口

**图5.6：**配置监听界面

* + 1. CFG写请求流（A路径）

Cadence IP内部有两个AXI-Lite主端口M0和M1，如顶层图[*5.6*](#_bookmark48)所示。

* + - * M0端口用于访问内部寄存器（A1路径）
      * M1端口用于访问外部配置寄存器（A2路径）

当接收到CFG写入时，内核将与TLP上接收的相同数据放置在M0和M1的写入地址和数据通道上。客户端可以使用M1上的写地址/数据通道来跟踪和监视任何IP CFG

由IP内部更新的寄存器M1上的写响应通道由控制器实现但不使用，使得针对写的返回完成响应纯粹基于M0端口，如下所示：

* + - * 当M0端口响应DECERR/SLVERR时，完成发送UR
      * 当M0端口响应OKAY（正常）时，完成将被发送为成功



配置请求

收到TLP

M0 AXI Lite写控制通道

M0 AXI Lite写入响应通道

CFG监听接口

M1 AXI Lite写控制通道

配置完成TLP

crt\_readycrt\_valid crt\_req

m0\_awreadym0\_awvalid m0\_awaddr

m0\_breadym0\_bvalid m0\_bresp

m1\_awreadym1\_awvalid m1\_awaddr

cct\_readycct\_valid cct\_cpl

W0 W1

A0 A1

R0

A0 A1

W2

A2

R1 R2

A2

C0 C1 C2

**图5.7：**配置监听接口写入

* + 1. CFG读请求流（A路径）

在配置读取期间，内核将地址置于M0和M1读取地址通道上客户端可以选择在M1上使用自己的读取数据而不是从内部寄存器返回的数据进行响应。内核使用来自M1的RRESP信号来决定在完成时选择发送哪个读取数据和响应。

当在入站上接收到CFG读取请求时，该请求被传递到M0和M1接口。只有在接收到来自M0和M1端口的响应后，才发送TLP

* + - * 当M1端口响应DECERR/SLVERR/REQ时，完成与来自M0的数据一起发送
      * 当M1端口响应OKAY（正常）时，完成将以Successful（成功）的形式发送，并带有来自M1的数据。如果用户未在M1上使用任何RRESP进行响应以进行读取，则总线可能挂起，并且配置路径可能停止。通过IP Core寄存器实现用户可编程超时机制，以确保配置路径永远不会卡住。



配置 CRT\_ready

请求crt\_valid

TLP crt\_req

R0

R1

R2

M0 AXI Lite m0\_arready

读控制 m0\_arvalid

通道 m0\_araddr

A0

A1

A2

m0\_rready

M0 AXI Lite

m0\_rvalid

读通道

m0\_rresp

R0

R1

R2

M1 AXI Lite m1\_arready

读控制 m1\_arvalid

通道 m1\_araddr

A0

A1

A2

m1\_rready

M1 AXI Lite

m1\_rvalid

读通道

m1\_rresp

R0

R1

R2

配置 cct\_ready

完成 cct\_valid

TLP cct\_cpl

C0

C1 C2

**图5.8：**配置监听接口读取

CFG监听接口

# 第6章：寄存器

第6章：**寄存器**

* 1. 寄存器描述

有关Register Description的信息，请参阅交付包的*<package\_path>/models/ipxact/\**目录

最多有三组寄存器描述，具体取决于配置：

* + - *\* \_ep\_local.xml/html*- 描述从Device AXI Lite接口看到的端点寄存器空间
    - *\* \_ep\_remote.xml/html*-描述从主机看到的远程端点寄存器空间
    - *\* \_rp\_local.xml/html*- 描述从主机AXI Lite接口看到的根端口寄存器空间
    - *\* \_rp\_remote.xml/html*- 描述从远程接口看到的根端口寄存器空间要从linux打开html文件，请键入：

<web\_browser\_name>文件名>.html

# 第7章：控制器和PHY集成解决方案

第7章：**控制器和PHY集成解决方案**

7.1控制器IP集成协议栈（IPS）

 注：如果此版本是控制器和PHY解决方案，则与本章相关。

控制器IP集成协议栈（IPS）将PCIe的控制器IP和PHY IP集成到单个可配置设计中。

IPS由围绕控制器IP和PHY IP的包装器组成。它包含与外部世界的顶级接口以及两个模块之间的互连IPS顶层的控制寄存器有少量逻辑该设计的框图如下图所示。

集成解决方案

物理顶层包装器

|  |  |
| --- | --- |
|  | 应用接口 |
| APB接口 | PCIe  控制器 |
|  | 管接口 |

|  |  |
| --- | --- |
|  | 管接口 |
| APB接口 | PCIe PHY  (PCS+ PMA） |
|  | 串行接口 |

**图7.1：**IPS框图

 注意：请参阅发行包*dbPcie\_ios\_top/hdl/hdl\_src/phy\_ip/<pcs>/doc中提供的PHY文档*

如上图所示，

* 控制器IP有一个应用程序接口，直接通过包装器连接
* 控制器IP通过PIPE接口连接到PHY IP，PIPE接口的大小取决于所选择的PHY
* PHY IP由数字PCS层和混合信号PMA组成，其串行接口通过封装器直接连接
* phy\_top\_wrapper实现了所有的捆绑和简单的逻辑，以保持*hdl/hdl\_src/ips/ cdnhpa\_ips.v的*整洁。

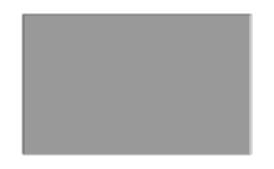
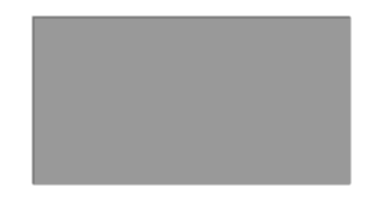
 注：有关如何在IPS级别运行仿真的信息，请参阅Cadence Verilog Testbench用户指南。

# 第8章：应用接口AXI4

第8章：**应用接口AXI4**

* 1. AXI概述

下图显示了控制器IP的概述，包括用于AXI4接口支持的AXI桥DMA（蓝色）和DTI-ATS（绿色）模块是可选的，仅在IP配置期间启用相应功能时才显示。禁用DTI-ATS功能时，AXI 4流接口不可用AXI4 DMA DS接口仅在DMA和单独的DMA描述符服务器端口都启用时可用。否则，DMA描述符服务器使用主AXI4管理器接口读取DMA描述符。



带AXI的

阿西大桥

AXI经理

CPCS/PL/DLL/TL/HLS RX

DMA

HLS RX

HLS

AXI4管理器

I/F

AXI

AXI4 DMA DS

I/F

PCIe串行

AXI下属

HLS

CPCS/PL/DLL/TL/HLS TX

HLS TX

AXI 4从属I/F

AXI4 Stream I/F

DTI-ATS

AXI

**图8.1：**带AXI的

* 1. AXI下属

AXI Subordinate接口使应用层能够作为总线主控器跨PCIe链路发起PCIe事务

对于端点（EP），只有当应用层具有总线主机功能时，此接口才必须连接到应用层

AXI下属发起请求并通过PCIe链路将其发送到完成者。经由区域或边带描述符的AXI写入和读取访问被解码并映射到相应的发布和非发布写入和读取TLP生成器以发起PCIe请求。存储器写入和消息（有/无数据）请求在Posted发生器中生成，并映射到Posted HLS TX接口。配置写入、IO写入、AtomicOp和DMWr请求在非发布写入生成器中生成，并映射到非发布HLS TX接口。存储器读取、配置读取、IO读取请求在非发布读取生成器中生成，并映射到非发布HLS TX接口。

* + 1. AXI从属操作类型

下表列出了应用程序层可以在AXI Subordinate上请求作为写/读操作的PCIe TLP类型更多信息可以在接下来的小节中找到。

表8.1：允许的AXI从属PCIe TLP类型

|  |  |  |
| --- | --- | --- |
| AXI操作 | 写入操作 | 读取操作 |
| 根端口（RP） | **MWr**内存写入请求  **AtomicOp**原子请求  **DMWr**可延迟内存写入请求**CfgRAM 0/1**Cfg写入类型0/1请求**IOWr**IO写入请求  消息请求  **VDM**供应商定义的消息请求 | **MRd**内存读请求**CfgRd 0/1**Cfg读类型0/1请求**IORd**IO读请求 |
| 终点（EP） | **MWr**内存写入请求  **AtomicOp**原子请求  **可**延迟存储器写请求  消息请求  **VDM**供应商定义的消息请求 | **MRd**内存读取请求 |

下表列出了每种类型的请求允许的传输类型更多信息可以在接下来的小节中找到这些限制是AXI和PCIe规范规则以及请求从AXI事务转换为PCIe TLP的结果。应用层可以在单个请求中使用允许的传输的组合

表8.2：每种请求类型允许的传输类型

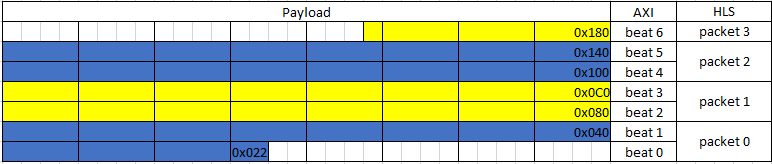
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 分动器类型 | 分路逻辑 | 绕转 | NCBE | 窄转移 |
| MWr -内存写入请求 | 是（以最大有效负载大小拆分） | 是的 | 是的 | 是的 |
| AtomicOp -AtomicOp请求  参见[*原子操作*](#_bookmark114)  更多详情 | 没有 | 否/  是（用于AXI-5比较和交换信号） | 没有 | 没有 |
| DMWr -可延迟内存写入请求 | 没有 | 没有 | 没有 | 是的 |
| Cfg 0/1 - Cfg写入类型0/1请求 | N/A | 没有 | 是的 | 没有 |
| IOWr - IO写请求 | N/A | 没有 | 是的 | 没有 |
| 消息/VDM -（供应商定义）消息请求 | 没有 | 没有 | 没有 | 是的 |
| MRd -内存读取请求 | 是（按最大读取请求大小拆分） | 是的 | 没有 | 是的 |
| CfgRd 0/1 - Cfg读取类型0/1请求 | N/A | 没有 | 没有 | 是的 |
| IORd - IO读取请求 | N/A | 没有 | 没有 | 是的 |

* + 1. AXI从属写入操作

AXI协议中的写事务是具有与相应通道相关联的独立地址、数据和响应阶段的拆分事务应用层必须遵循*AXI规范*中描述的写地址、数据和响应通道的管理器协议。作为IP的一部分，AXI从属应遵循*AXI规范*中所述的写入地址、数据和响应通道的从属协议。

应用层可以在AXI上发送具有大于当前PCIe最大有效载荷大小（MPS）的有效载荷的存储器写请求AXI Subordinate处理将请求拆分为多个PCIe内存写入TLP，其长度不超过MPS。第一次拆分从与数据路径宽度对齐的地址开始。

下图显示了MPS设置为128字节和512位（64字节）数据路径的示例。AXI数据包的第一次拆分发生在AXI的前两个节拍之后-字节从与数据路径对齐的地址（示例中为0x 00）开始计数，第二次分裂发生在下两个完整的数据节拍之后，依此类推。



**图8.2：**AXI从属存储器写请求拆分

如果禁用了窄事务支持，则当请求中的*axi\_s\_awlen*不为零时（例如，对于多拍访问），不允许axi\_s\_awsize偏离其数据路径大小的最大值

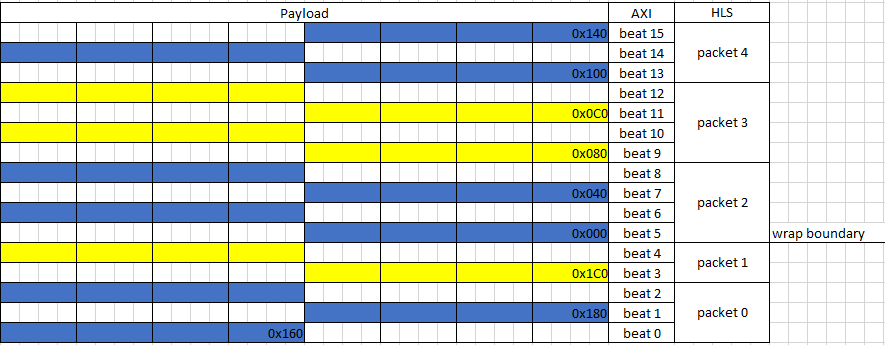
启用窄访问支持时，可以发送内存请求，其中*axi\_s\_awsize*不等于任何有效*axi\_s\_awlen*（单节拍或多节拍事务）的数据路径宽度的下图显示了使用窄访问和大于MPS值的有效负载大小通过AXI发送的请求。



**图8.3：**AXI从属存储器写入窄请求拆分

AXI Subordinate不支持FIXED突发类型。如果应用层需要将数据发送到同一地址，则应将数据拆分为具有相同地址的INCR突发类型的多个单拍（*axi\_s\_awlen*

如果应用层发送具有WRAP突发类型的请求，则事务在包装边界地址处被拆分下图显示了使用窄访问、WRAP突发类型和大于MPS值的有效负载大小通过AXI发送的请求



**图8.4：**AXI从属存储器写入窄回绕请求拆分

配置和I/O（非发布）写入的处理方式类似，只是数据有效载荷的长度只有一个DWord。核心仅在从PCIe链路接收到完成返回时才发出配置和I/O写入响应

* + - 1. 可延迟内存写入（DMWr）

可延迟内存写入是PCIe非发布写入请求。该请求需要PCIe完成（无数据），其携带状态，该状态是完成者对该请求的响应。这为完成者提供了一种接受请求（成功完成状态）或暂时拒绝（推迟）请求（请求重试状态）的机制。DMWr数据预计将在单个请求中作为一个整体进行传输（并被完成者接受/拒绝），因此由于各种条件（小于MPS，无WRAP突发类型，无NCBE），它无法拆分。

DMWr请求的访问和生成在[*AXI从属访问*](#_bookmark71)部分中描述。

* + - 1. AXI从属写入ID管理

AXI Subordinate接口上的写请求作为拆分事务执行。也就是说，客户端可以在对写入请求的响应未决时做出附加的读取和写入请求。每个未完成的写请求可以由客户端使用相同的*axi\_s\_awid*或唯一的axi\_s\_awid发出。对于非发布写入，AXI Subordinate在内部将*axi\_s\_awid*映射到内部生成的PCIe标记。有关详细信息，请参阅：[*未过帐事务处理的标记管理*](#_bookmark62)查找该映射以将传入的完成转换为与写入响应通道一起发送的对应的*axi\_s\_bid*值

* + - 1. 零长度写入

零长度内存写入事务可以在AXI Subordinate接口上启动，方法是按照1字节内存写入事务的情况进行处理，但*axi\_s\_wstrb*中的字节有效位在数据周期期间全部设置为零除外。这导致核在PCIe链路上发送存储器写入请求，其中长度字段被设置为1个双字，并且字节使能字段被设置为全零。

* + - 1. 非连续字节启用写入

PCIe规范允许在单DWord写入时使用非连续字节使能（NCBE）进行内存写入，并且当地址在8字节边界上对齐时，允许在双DWord写入时使用NCBE进行内存写入。对于这些写入事务，用户逻辑基于请求TLP的报头中指示的有效字节来设置*axi\_s\_wstrb*信号上的字节有效位

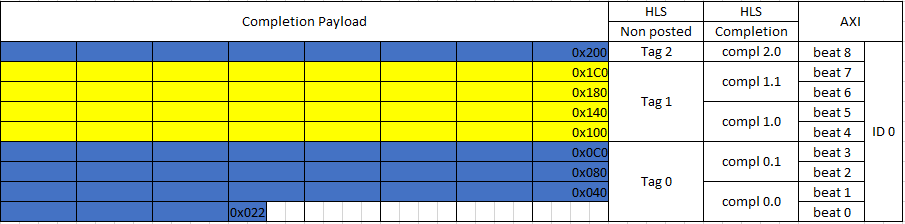
对于单个DWord的AXI写入，字节有效可以是不连续的，如PCIe规范所允许的。同样，对于2-DWord写入，如果起始地址在偶数DWord边界上对齐，则字节有效可以是不连续的

对于负载大于2DWord的AXI写入，请求被分成多个2DWord NCBE PCIe写入请求。否则，禁止发送有效字节不连续且有效负载大于两个DWord的事务。

* + 1. AXI从属读操作

AXI协议中的读事务是具有与相应通道相关联的独立地址和数据阶段的拆分事务。客户端必须遵循*AXI规范*中所述的读取地址和数据通道的管理器协议。AXI Subordinate遵循*AXI规范*中描述的读地址和数据通道的Subordinate协议。

用户逻辑可以在AXI上发送具有大于PCIe中允许的最大读取请求大小的有效载荷的存储器读取请求。AXI Subordinate将请求拆分为多个PCIe内存读取TLP。该请求拆分为具有*最大读取请求大小*（MRRS）的请求，其中第一个拆分从与数据路径对齐的地址开始计数下图显示了MRRS设置为256字节和512位数据路径的示例。标记0的第一次分割读取AXI上的前四个节拍，即字节计数从与数据路径对齐的地址（此处为0x 00）开始，直到在AXI上读取256个字节（即4个节拍）第二次分裂发生在四个完整的有效数据节拍之后，依此类推。



 注意事项：

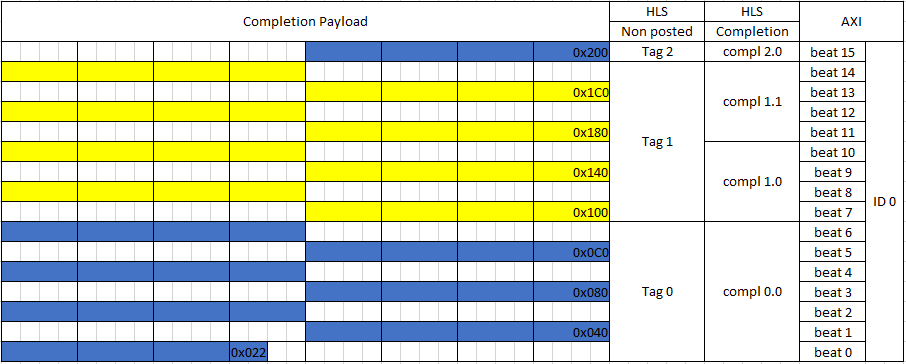
**图8.5：**AXI从属内存读取请求拆分

来自具有标签1和2的拆分请求的完成可以在来自标签0的完成之前到来，但是它们的有效载荷被保存并且仅在来自标签0的完成的有效载荷通过PCIe链路到来之后才被放在AXI上。也就是说，数据以增加的地址顺序通过AXI发送以遵守AXI规则。

 注意事项：

如果禁用窄事务，则当请求中的*axi\_s\_arlen不为零时，*不允许*axi\_s\_arsize*内存请求可以通过窄访问发送，即*axi\_s\_arsize*不等于任何*有效axi\_s\_arlen*的数据路径大小的最大值。

下图显示了使用窄访问和大于MRRS值的有效负载大小通过AXI发送的请求



**图8.6：**AXI从属内存读取窄化请求拆分

如果用户逻辑发送具有WRAP突发类型的请求，则事务在包装边界地址处被拆分。下图显示了使用窄访问、WRAP突发类型和大于MRRS值的有效负载大小通过AXI发送的请求



**图8.7：**AXI从属内存读取窄行绕回请求拆分

在根端口中，AXI Subordinate接口还用于发起配置和I/O读取请求。配置和I/O读取的工作方式与内存读取相同。这些请求通过transaction-type字段与内存请求区分开。响应这些请求返回的数据总是4字节长，并与请求地址对齐。

* + - 1. AXI从属读取ID管理

AXI Subordinate接口上的读取请求作为拆分事务执行。也就是说，客户端可以在读取请求的完成未决时做出附加的读取和写入请求。每个未完成的读请求可以由客户端使用相同的*axi\_s\_arid*或唯一的axi\_ s\_arid发出。AXI Subordinate在内部将*axi\_s\_arid*映射到内部生成的PCIe标签。有关详细信息，请参阅：[*未过帐事务处理的标记管理*](#_bookmark62)查找此映射以将传入的完成转换为与读取数据通道一起发送的对应*axi\_s\_rid*值

* + - 1. 完工数据排序

AXI规范规定，如果存在具有相同*axi\_s\_arid的*未完成读取请求，则读取数据应按顺序由AXI Subordinate接口返回。然而，由于这些未完成的读取请求中的每一个都在PCIe侧被分配了唯一的标签，因此PCIe排序规则允许读取完成以任何顺序返回。AXI下属重新排序无序完成，并以正确的顺序在AXI下属读取数据通道上发出。如果客户端逻辑发出具有唯一*axi\_s\_arid*的读取请求，则AXI排序规则允许读取数据无序返回，并且AXI从属以读取数据从PCIe链路返回的顺序发出读取

* + - 1. R通道数据交错

数据交织可以发生在AXI的R信道假设两个AXI读取请求具有AXI ID 'h11和'h22。具有AXI ID 'h11的请求具有等于'h0A的ARLEN，这意味着11个节拍。11 \* 64 b = 704字节，超过MRRS限制的512字节。因此，AXI Subordinate将此AXI请求拆分为两个PCIe存储器读取，以避免MRRS违规。

* + - * + 一个TAG 'h000的长度为128 DW（512字节）。
        + 另一个TAG 'h001的长度为48 DW（192字节）。

具有AXI ID“h22”的请求具有等于“h05”的ARLEN，其为6个节拍。这不超过512字节的MRRS限制。因此，该AXI请求被转换成单个PCIe存储器读取，其被分配TAG 'h002。具有不同TAG的PCIe请求的完成假设三个PCIe请求按以下顺序完成：

1. TAG 'h000（轴ID'h11）
2. TAG 'h002（AXI ID'h22）
3. TAG 'h001（AXI ID'h11）

它是在AXI R信道上发送有效载荷的顺序。也就是说，来自具有标签“h002”的AXI ID“h22”的数据在具有标签“h000”的AXI ID“h11”的数据和具有标签“h001”的AXI ID“h11”的数据之间交错。AXI从属块仅跟踪具有特定AXI ID内的不同PCIe TAG的事务的顺序因为TAG 'h002属于不同的AXI ID，所以即使与AXI ID 'h11（TAG h'000和'h001）的事务尚未与RLAST结束，它也呈现在AXI从属R通道上。

在上面示例的情况下，可以使用以下选项

* + 在应用层处理AXI从属R通道数据交织
  + 避免将AXI事务拆分为多个PCIe内存读取请求
    - 通过使用ARLEN = MRRS，例如在这种情况下，最大ARLEN是'h 07，它指示64字节-512字节的8个节拍
    - 通过增加MRRS，例如，到1024字节，因此ARLEN 'h0A（直到'h0F）不会导致拆分
  + 对AXI交易
    - AXI Subordinate跟踪单个AXI ID池中PCIe TAG的顺序。在所描述的情况下，它将呈现TAG 'h000的数据（没有RLAST），保留TAG'h002的数据，呈现TAG 'h001的数据（具有RLAST，因为只有第一个AXI读取结束），当它被接收时，并且只有然后呈现TAG'h002的数据
    1. 未过帐事务处理的标记管理

AXI Subordinate维护由应用层发起的所有未决非发布事务（内存读取、I/O读取和写入、配置读取和写入以及可延迟内存写入）的状态，以便完成器返回的完成可以与相应的请求相匹配。每个未结算的未过帐交易的状态保存在AXI下属接口模块的AXI下属完成LUT中它的深度决定了未结算的未过帐事务处理的最大数量。分配给非发布PCIe请求的标签池的偏移量和范围由非发布请求寄存器的标签管理设置（有关更多详细信息，请参见寄存器描述每个启动的标签对于每个待处理的未发布的

请求过程中返回的完成与挂起的请求使用它们的标签进行匹配AXI Subordinate Completion LUT维护一个自由标签列表，并在应用层启动非发布事务时将其中一个标签分配给每个请求。

 注意事项：

AXI下级完成LUT具有所有请求通用的单个标签池。它由单个偏移量和范围定义，这意味着不能同时使用不同的PCIe标签范围（0-31、0-255、256-1023、1024-16383）

要使用10位标签（256-1023）功能，应用层应针对所有功能设置设备控制2寄存器中的10位标签验证器使能位

要使用14位标签（1024 - 16383）功能，应用层应在设备控制3寄存器中为所有功能设置14位标签验证器启用位。请注意，即使14位标签能力可用，可用标签的数量仍然受到AXI从属完成LUT深度的限制。

如果在任何功能中，设备控制寄存器中的扩展标签字段使能位被清除，因此10位和14位标签校验器使能位也应该被清除，则应用层应该为AXI从属完成LUT设置5位标签范围（0-31），作为公共标签池的结果。

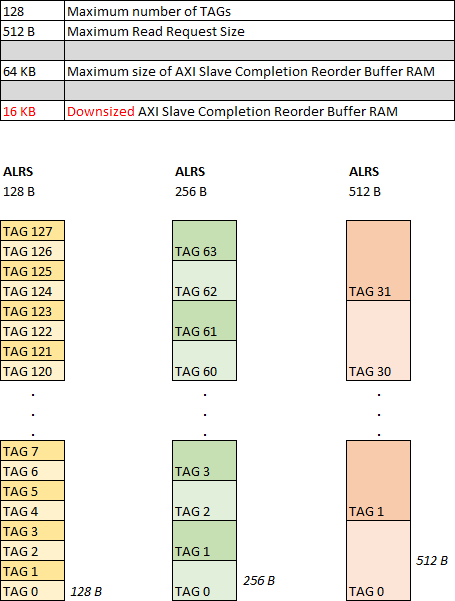
* + 1. 应用层读取大小

默认情况下，AXI从属完成重新排序缓冲区RAM大小等于可用PCIe TAG的最大数量（AXI从属完成LUT深度）乘以最大支持MRRS。例如，64 KB用于128个TAG和512 B MRRS。

 注意：ECC保护开销不包括在此计算中。

对于大量TAG和大MRRS，这样的完成重排序缓冲器RAM大小可能太大并且不适合控制器使用情况。例如，当应用层想要发送许多具有较小大小的读请求（利用最大数量的TAG），但有时更少的读请求，但具有更大的大小。

在初始IP配置过程中，可以缩小AXI从属完成重新排序缓冲区RAM的AXI从属*axi\_slave\_cnc*寄存器的应用层读取大小字段可用于在控制器操作期间选择每个TAG块的完成重排序缓冲器RAM的所需大小。



**图8.8：**AXI从属应用层读取大小

[*图8.8所*](#_bookmark65)示为各种ALRS设置下的AXI次级完成重排缓冲RAM的布局。当ALRS等于128 B时，可以使用最大数量的TAG当ALRS设置为较大值时，可用TAG的数量会减少，因为没有足够的内存用于PCIe完成中返回的数据-分别有64个和32个TAG可用于256 B和512 BALRS。

 注：应用层负责通过AXI下属*的范围*字段限制可用TAG的数量

*tag\_management*相应地防止完成重新排序缓冲区RAM溢出。

大于ALRS的读请求仍然可以由应用层发出，因为它们将被AXI从属块分成不违反ALRS限制并适合保留的完成重排序缓冲区RAM块的较小TLP

* + 1. PF/VF编码

PF/VF（ipfivf）编码用于AXI元数据和其他控制器接口。

PCIe控制器允许将**SR\_IOV**步幅设置为2的幂的数字，作为可配置选项。连接到PF的VF的数量也可以在链路训练之前使用本地管理寄存器动态地修改

ipfivf格式给出了附加PF中的VF编号和关联的PF编号。这有助于客户端逻辑使用*sriov\_stride*和分配给PF的VF解码BDF请求者ID格式。

PF也以相同的格式编码，其中一个比特指示编码是否用于VF。

**VF\_STRIDE**的值大于1，只有在需要多个总线号来分配VF时才有用。当所有PF都分配了相同数量的总VF且主机可能编程较小时，**VF\_STRIDE**的2次幂非常有用

*numVFs*用于优化总线号的使用如果*total\_vf*是基于应用程序分布的，并且主机不太可能优化总线号，则步幅为1是好的

仅支持步长值1和大于PF数的2次方数

[*表8.3*](#_bookmark67)显示了使用不同步幅和VF分配到PF的ipfivf编码示例

表8.3：首字母缩略词和描述

|  |  |
| --- | --- |
| 缩写 | 描述 |
| 摆脱 | 与PCIe TLP |
| 国际石油天然气联合会 | 在关联PF |
| 总vf | 分配给PF的总VF，可通过LM寄存器编程。这是**SRIOV**只读功能字段。 |
| num\_vf | 由主机编程**的SRIOV**功能中的VF计数启用数量 |
| vf偏移 | **SRIOV**功能中的RO字段。（*vf\_offset*+ PF的RID）给出PF中第一个VF的RID。当*vf\_stride*不为1时，vf\_offset与vf\_stride相同当*vf\_Stride*为1时，*vf\_offset*为所有先前PF的（*num\_pfs*+（total\_vfs-1）） |
| vf\_Stride | **SRIOV**功能中的RO字段VF的*RID + vf\_Stride*给出PF中下一个VF的RID |



**图8.9：**颜色代码

* + - 1. 具有统一数量的总VF的示例

每个PF中有4个VF，主机仅启用了PF 0中的两个VF

|  |  |  |  |
| --- | --- | --- | --- |
| PF总数：2，VFS总数：每个PF 4，PF0.num\_vf=2，PF1.num\_vf=4：vf\_stride=1 | | | |
| 摆脱 | 国际石油天然气联合会 | | |
| BDF格式 | 相关PF | PF中的VF数量 | Is\_VF |
| 0 | 0 | - | 0 |
| 1 | 1 | - | 0 |
| 2 | 0 | 0 | 1 |
| 3 | 0 | 1 | 1 |
| 4 | 0 | 2 | 1 |
| 5 | 0 | 3 | 1 |
| 6 | 1 | 0 | 1 |
| 7 | 1 | 1 | 1 |
| 8 | 1 | 2 | 1 |
| 9 | 1 | 3 | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| PF总数：2，VFS总数：每个PF 4，PF0.num\_vf=2，PF1.num\_vf=4：vf\_stride=2 | | | |
| 摆脱 | 国际石油天然气联合会 | | |
| BDF格式 | 相关PF | PF中的VF数量 | Is\_VF |
| 0 | 0 | - | 0 |
| 1 | 1 | - | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
| PF总数：2，VFS总数：每个PF 4，PF0.num\_vf=2，PF1.num\_vf=4：vf\_stride=2 | | | |
| 2 | 0 | 0 | 1 |
| 3 | 1 | 0 | 1 |
| 4 | 0 | 1 | 1 |
| 5 | 1 | 1 | 1 |
| 6 | 0 | 2 | 1 |
| 7 | 1 | 2 | 1 |
| 8 | 0 | 3 | 1 |
| 9 | 1 | 3 | 1 |

 注：在上表中，***突出显示的条目***具有禁用/未使用的VF实例。

* + - 1. 总VF数不均匀的示例

PF 0中有2个VF，PF 1中有6个VF可用主机仅在PF 1中启用了四个VF，在PF 0中仅启用了一个VF

|  |  |  |  |
| --- | --- | --- | --- |
| PF总数：2，Pf0.total VF = 2，PF1.total VF=6，PF0.num\_vf=1，PF1.num\_vf=4，步幅=1 | | | |
| 摆脱 | 国际石油天然气联合会 | | |
| BDF格式 | 相关PF | PF中的VF数量 | Is\_VF |
| 0 | 0 | - | 0 |
| 1 | 1 | - | 0 |
| 2 | 0 | 0 | 1 |
| 3 | 0 | 1 | 1 |
| 4 | 1 | 0 | 1 |
| 5 | 1 | 1 | 1 |
| 6 | 1 | 2 | 1 |
| 7 | 1 | 3 | 1 |
| 8 | 1 | 4 | 1 |
| 9 | 1 | 5 | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| PF总数：2，Pf0.total VF = 2，PF1.total VF=6，PF0.num\_vf=1，PF1.num\_vf=4，步幅=2 | | | |
| 摆脱 | 国际石油天然气联合会 | | |
| BDF格式 | 相关PF | PF中的VF数量 | Is\_VF |
| 0 | 0 | - | 0 |
| 1 | 1 | - | 0 |
| 2 | 0 | 0 | 1 |
| 3 | 1 | 0 | 1 |
| 4 | 0 | 1 | 1 |
| 5 | 1 | 1 | 1 |
| \* \* | - | - | - |
| 7 | 1 | 2 | 1 |
| \* \*\* | - | - | - |

|  |  |  |  |
| --- | --- | --- | --- |
| PF总数：2，Pf0.total VF = 2，PF1.total VF=6，PF0.num\_vf=1，PF1.num\_vf=4，步幅=2 | | | |
| 9 | 1 | 3 | 1 |
| 十 \* | - | - | - |
| 11 | 1 | 4 | 1 |
| 10 \* | - | - | - |
| 13 | 1 | 5 | 1 |

 注意事项：

* + - * + 在上表中，***突出显示***的行具有禁用/未使用的VF实例。
        + 在上表中，***带有 \****的行具有无效VF。
    1. AXI从属访问

有两种方法可以执行AXI Subordinate访问

* 静态方法：基于
  + 当您只有TLP类型（读/写）并希望通过PCIe在出站方向上发送预编程的TLP信息时，基于区域的方法非常有用存储在区域中的TLP信息的任何更改都必须通过APB/AXI Lite重新编程完成。
* 动态方法：基于
  + 当您已经将TLP信息存储在控制器外部并且希望通过PCIe以出站方向发送时这将绕过地址

PCIe AXI中的转换逻辑已经拥有TLP信息的客户可以使用此功能，以便在需要更改TLP信息时节省重新编程的时间。（如果TLP信息在基于区域的方法中发生变化，则需要按区域对PCIe描述符寄存器进行

* + 1. AXI通过区域进行

PCIe核心的从属地址空间可以划分为多个区域。一次可以有最多的区域活动，每个区域都有一组寄存器用于其操作。出站访问是一个两步过程

* 区域设置

区域寄存器需要在访问它们之前进行编程区域寄存器可以使用提供的APB/AXI Lite接口进行编程

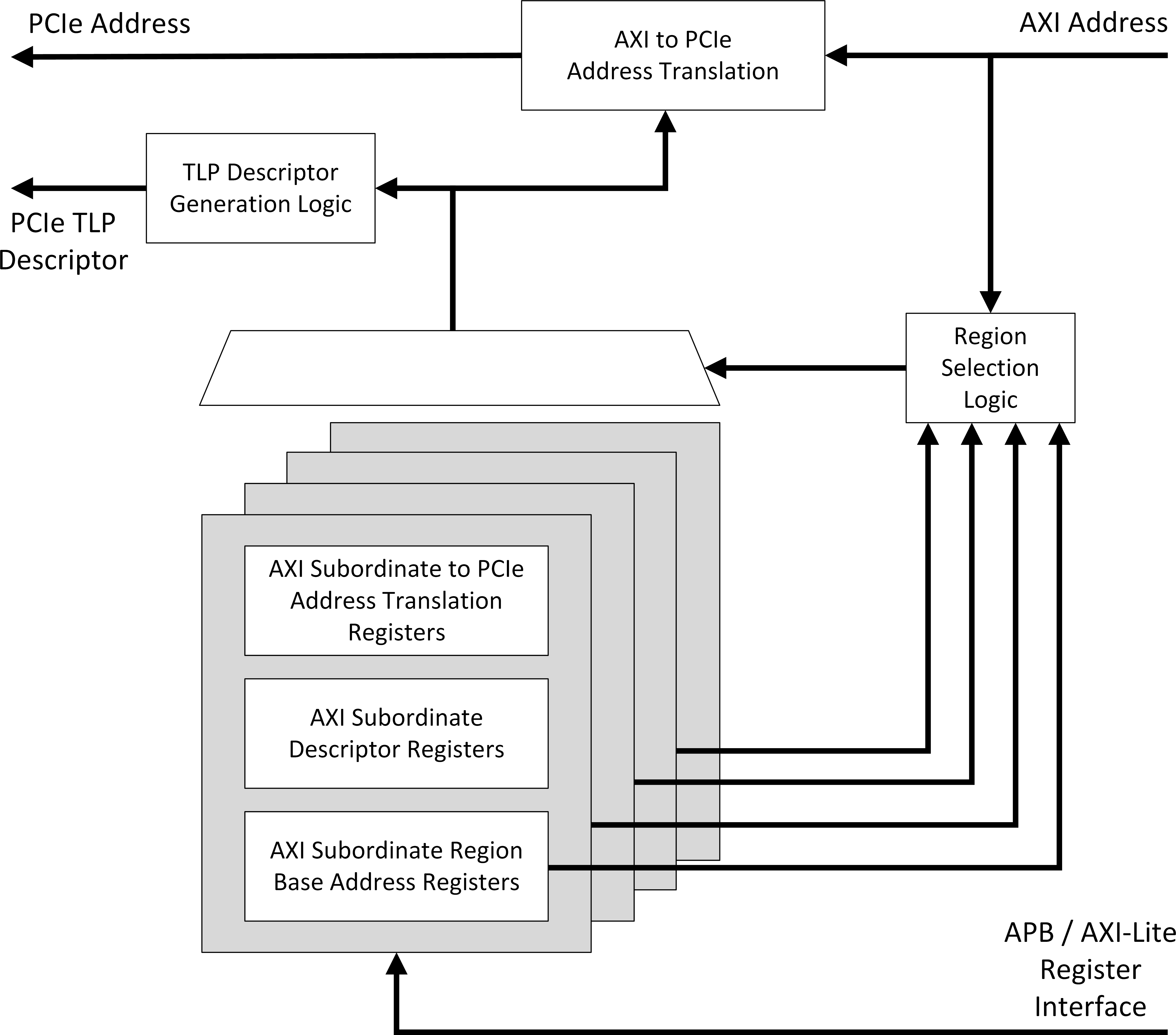
* 区域访问

区域寄存器可以通过AXI Subordinate接口（AXI Subordinate或axi\_s\_\*）访问

每个区域都有以下一组寄存器。

表8.4：一个区域的AXI从属区域访问寄存器

|  |  |
| --- | --- |
| 寄存器集名称 | 描述 |
| AXI从属于PCIe地址转换寄存器 | 此寄存器组用于执行从AXI地址到PCIe地址的地址转换 |
| AXI从属描述符到TLP寄存器 | 此寄存器组保存要发送到出站的TLP信息 |
| AXI从属区域基地址寄存器 | 该寄存器组保存AXI区域基地址和相应区域的区域大小。该寄存器组用于通过匹配传入的AXI地址对区域进行解码。 |



**图8.10：**PCIe控制器AXI通过区域进行从属访问

* + - 1. AXI从属于PCIe地址转换寄存器

这些寄存器用于为内存和IO TLP执行从PCI-AXI到PCIe的地址转换。如果通过位被编程为通过少于特定位，则配置TLP总线号、器件号和功能号可在PCI-AXI至PCIe地址转换寄存器请参阅交付包的*<package\_path>/models/ipxact/\**目录中的

一些AXI从属地址（*axi\_s\_awaddr*）位用于传送消息TLP的 MSG\_ROUTING和

文件*<package\_path>/docs/cdnhpa\_ips\_axi\_s\_region\_access\_examples.html*包含通过区域进行的各种AXI从属访问的示例，从而导致不同的PCIeTLP。

* + - 1. AXI从属描述符寄存器

这些寄存器保存TLP中的静态信息（功能编号、加密器ID等）。参考交付包的*<package\_path>/models/ipxact/\**目录中的

文件*<package\_path>/docs/cdnhpa\_ips\_axi\_s\_region\_access\_examples.html*包含通过区域进行的各种AXI从属访问的示例，从而导致不同的PCIeTLP。

* + - 1. AXI区域基址寄存器

PCIe核心使用区域选择逻辑来匹配出站AXI地址和预编程的AXI地址（在每个区域的AXI区域基址寄存器中），以找出它所属的区域。匹配从区域0到最大区域数减1。比较器选择第一个匹配区域作为区域编号，该区域编号用于拾取静态TLP信息（PCIe描述符）和PCIe地址（用于地址转换）。AXI区域大小和区域基地址可以通过这种方式进行编程请参考交付包的*<package\_path>/models/ipxact/\**目录中的AXI区域基址

 注意事项：

不支持重叠区域。

所有AXI区域的起始地址与区域大小对齐，区域大小通过AXI区域基址寄存器axi\_addr 0 [5：0]编程。如果区域选择逻辑没有找到匹配，则在AXI处返回适当的错误响应

* + 1. 通过边带描述符的

如果用户想直接发送PCIe地址包而不进行任何地址转换，可以使用此方法这在动态地址转换的情况下很有用。

在这种AXI从属访问方案中，PCIe地址直接在*axi\_s\_aw/raddr*驱动。在该方案中，不在*axi\_s\_aw/ruser*中直接驱动PCIe描述符。有一个VALID位用于启用边带访问，即*axi\_s\_aw/ruser*中的第159位，它使边带访问方案优先于区域访问方案。

关于边带描述符，请参考交付包的*<package\_path>/docs/cdnhpa\_ips\_IO\_configtions. html*文件中的axi\_s\_awuser/axi\_s\_aruser信号描述

文件*<package\_path>/docs/cdnhpa\_ips\_axi\_s\_sideband\_access\_examples.html*包含通过边带描述符进行的各种AXI从属访问的示例，这些访问导致不同的PCIeTLP。

* + 1. AXI下级访问说明
       - * 在AXI Subordinate接口上发出请求之前，客户端逻辑必须检查以下条件

只有根端口可以启动I/O和配置。

源函数当前未处理函数级复位（当在内核中实现FLR

* + - * + AXI从属期望客户端逻辑对[*AXI从属描述符寄存*](#_bookmark74)器进行编程，根据该AXI从属描述符寄存器生成符合PCIe规范的TLPs。
        + AXI Subordinate预期正常/供应商定义消息请求的数据必须与AXI传输中数据通道（*axi\_s\_wdata*）上的DWORD（4字节）对齐
        + 不带数据的消息不应设置字节选通（*axi\_s\_wstrb*= 0）。
        + AXI Subordinate支持多个未完成的ATS翻译请求。但是，每个请求只允许请求一次翻译（AT请求TLP的长度必须限制为2个DWORD，8个字节）。ATS转换请求的AXI地址必须与8个字节对齐。
        + AXI下属公司不允许使用相同的AXI ID错误地发出发布和未发布的写请求，因为它不能保证在这种情况下AXI B响应的顺序。

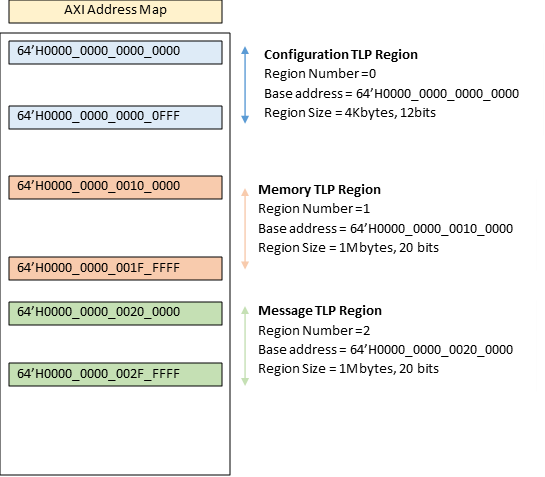
如果应用层想要使用相同的AXI ID来发送已发布和未发布的写请求，则它必须限制发出另一种请求，直到对第一种请求的所有AXI B响应返回

* + - * + AXI Subordinate在访问后阻止所有新请求，从而导致解码错误（DECERR），直到它处理所有未完成的请求。这可以防止违反AXI响应顺序。
        + 当启用*ecrc\_capability\_support*时，ECRC由控制器IP在内部生成应用层

必须确保ECRC生成使能位（高级错误能力和控制寄存器）为给定功能设置

* + - * + **PCIe 6，微片模式**- 如果应用层生成的写请求可能具有非连续字节启用，则必须为此类请求
    1. AXI通过示例实现

下图显示了标记了不同类型区域的CAMAXI内存映射。该映射仅显示了将区域划分为3个，并具有相应的基址和区域大小。但是，一旦我们使用中给出的每种类型TLP的正确值对“AXI从属描述符到TLP寄存器”进行编程，就可以分配将基址和区域大小编程到“AXI区域基址寄存器”中，以获得相应的区域编号。



**图8.11：**TLP访问的AXI从属地址映射

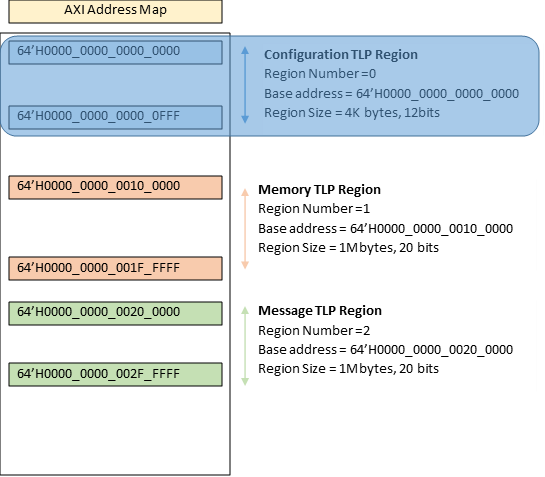
在本例中，下表给出了“AXI区域基址寄存器”的编程值

表8.5：AXI区域基址寄存器的值

|  |  |  |  |
| --- | --- | --- | --- |
| 地区号 | axi\_addr 0 [5：0] | [31：08] | axi\_addr1[31：0] |
| 0（配置TLP） | 6'd11 | 24小时00\_0000 | 32'h0000\_0000 |
| 1（内存或IO TLP） | 6'd19 | 24小时00\_1000 | 32'h0000\_0000 |
| 2（消息TLP） | 6'd19 | 2000年00时24分 | 32'h0000\_0000 |

* + - 1. 配置TLP访问

有两种方法可以传递配置TLP的BDF（完成者ID）信息配置TLP始终被路由到EP设备的PCIe配置空间。



**图8.12：**配置TLP的AXI从属地址映射

**方法1**：总线号、设备号和功能号通过“AXI从属于PCIe地址转换寄存器

配置寄存器地址可通过axi\_s\_aw/araddr[11：0]位驱动。应将“AXI从属于PCIe地址转换寄存器”的通过位或addr 0 [5：0]编程为6“d11，使得较低的12”d11“可被编程

位取自axi\_s\_aw/araddr。在这种情况下，BDF信息被编程在对应区域（在该示例中为区域0）的“PCIe地址转换寄存器的AXI从属”中，而扩展寄存器号和寄存器号从axi\_s\_aw/araddr[11：0]位传递。

表8.6：通过出站AXI至PCIe地址转换寄存器对配置TLP进行BDF值编程（方法1）

|  |  |  |  |
| --- | --- | --- | --- |
| 传统模式 | | ARI模式 | |
| addr 0位 | 值 | addr 0位 | 值 |
| 二十七点二十分 | 总线编号 | 二十七点二十分 | 总线编号 |
| 十九点十五分 | 器械编号。 | 十九点十二分 | 功能号 |
| 十四点十二分 | 功能号 |

上表显示了传统和ARI模式下“AXI从属于PCIe地址转换寄存器”的addr 0寄存器中要编程的BDF值

 注意事项：

如果我们使用这种发送配置TLP的方法，AXI区域大小至少应为4K字节

**方法2**：通过axi\_s\_aw/araddr的总线号、设备号和功能号

在这种发送配置TLP的模式下，我们可以将“AXI从属于PCIe地址转换寄存器”的addr 0 [5：0]编程为6'd27，以便将AXI地址的低28位直接传递到PCIe内核这些位包括如下所示的BDF值。

表8.7：通过PCI-AXI到PCIe地址转换寄存器的配置TLP的BDF值编程（方法2）

|  |  |  |  |
| --- | --- | --- | --- |
| 传统模式 | | ARI模式 | |
| axi\_s\_aw/araddr位 | 值 | axi\_s\_aw/araddr位 | 值 |
| 二十七点二十分 | 总线编号 | 二十七点二十分 | 总线编号 |
| 十九点十五分 | 器械编号。 | 十九点十二分 | 功能号 |
| 十四点十二分 | 功能号 |

 注意事项：

由于BDF信息通过axi\_s\_aw/araddr或AXI地址的27：12传递，因此在此方法中，AXI区域大小应至少为256 Mbytes（28位）

在这两种方法中，addr 1 [31：0]应编程为零。

* + - 1. AXI从属描述符到TLP寄存器的编程

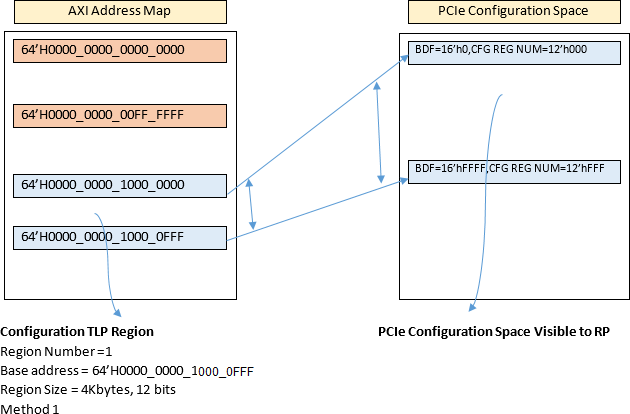
AXI从属描述符寄存器编程一旦完成，就将特定的TLP类型分配给区域编号。对于本例，请确保您遵循区域0 desc0、desc1等中要编程的值中给出的寄存器

*<package\_path>/models/ipxact/\**目录。

配置TLP AXI到PCIe地址转换

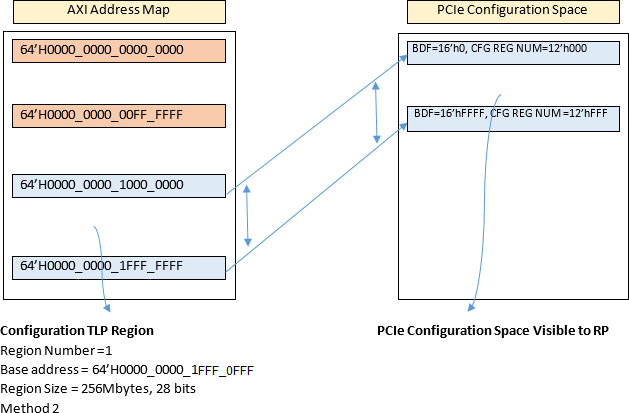
考虑一个不同的示例，其中AXI配置TLP地址映射不是从64'h0000\_0000\_0000\_0000开始，然后我们可以使用PCIe内核中的地址转换功能将地址转换为请求者id和配置寄存器号。这可以被认为是“对RP可见的PCIe配置空间

**方法1**：在该方法（方法1）中，AXI区域可以具有4K字节的区域大小（AXI地址的低12位BDF信息是从区域寄存器中捕获的，如上面的方法1描述中所解释的。



**图8.13：**使用方法1的配置TLP的AXI从属于PCIe配置空间地址转换

**方法2**：在该方法（方法2）中，AXI地址的低32位用于传送配置寄存器地址和BDF信息。



**图8.14：**使用方法2的配置TLP的AXI从属于PCIe配置空间地址转换

 注：PCIe配置写入/IO写入请求的有效负载大小为1 DW。

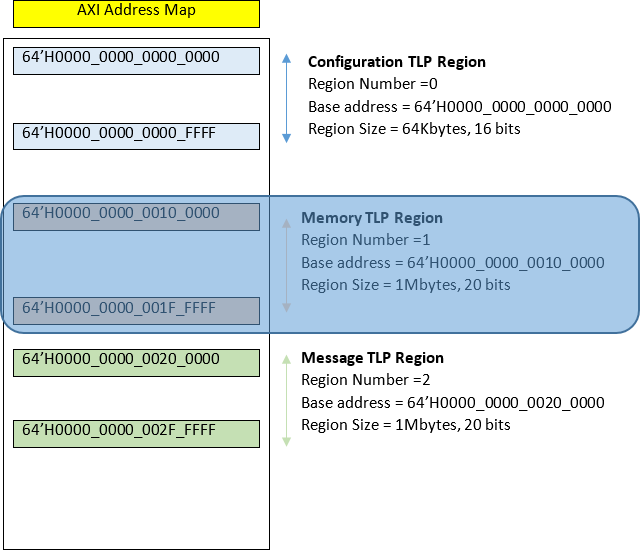
1. AXI地址应与提供的第一个数据字节使能对齐Data/BE从给定的第一个地址位置开始
2. 从上述地址开始的一个数据字用于配置/IO写入有效负载。

表8.8：PCIe配置写入/IO写入请求示例

|  |  |  |
| --- | --- | --- |
|  | Example1 | Example2 |
| axi\_s\_address | 0x4 | 0x0 |
| axi\_s\_wstrb | 0xF0 | 0x0F |

* + - 1. 内存或IO TLP访问

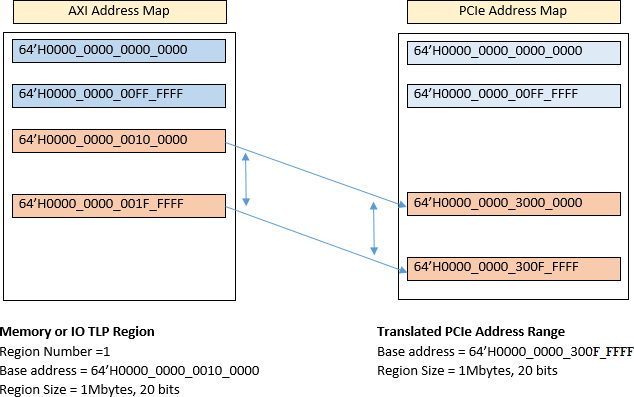
让我们假设PCIe控制器的存储器区域中的一个具有AXI区域基地址为64'h0000\_0000\_1000\_0000区域1，如下图所示），区域大小为1M字节。例如，假设要写入的PCIe基址是64'h0000\_0000\_3000\_0000。



**图8.15：**AXI存储器或IO TLP访问的从属地址映射

AXI从属描述符到TLP寄存器的编程

AXI从属描述符寄存器编程一旦完成，就将特定的TLP类型分配给区域编号。确保遵循交付件中提供的desc0、desc1等寄存器中的编程值。内存或IO TLP列。



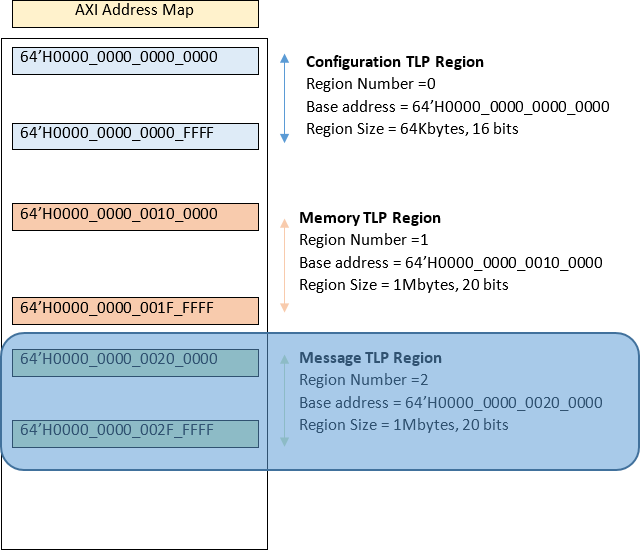
**图8.16：**AXI从属于PCIe地址转换寄存器编程

表8.9：AXI区域基址寄存器和PCIe地址转换寄存器值的AXI从属

|  |  |  |  |
| --- | --- | --- | --- |
| 寄存器 | 值 | 寄存器 | 值 |
| axi\_addr 0 [5：0] | 6'd19 | 地址0 [5：0] | 6'd19 |
| [31：08] | 24小时00\_1000 | 地址0 [31：8] | 24小时30\_0000 |
| axi\_addr1[31：0] | 32'h0000\_0000 | 地址1 [31：0] | 32'h0000\_0000 |

消息TLP访问

当对AXI区域进行PCIe消息访问时，PCIe报头字段主要从区域寄存器值驱动只有消息代码和消息路由字段通过AXI地址*axi\_s\_awaddr*驱动。



**图8.17：**AXI用于消息TLP访问的从属地址映射

此示例使用区域2作为消息区域。一旦利用落在64'h0000\_0000\_0020\_0000和64'h0000\_0000\_002F\_FFFF之间的AXI地址进行访问，区域2的**AXI从属描述符寄存器**需要使用消息TLP的值进行编程。

 注意事项：

消息的最小区域大小应为512 Kbytes（消息代码、消息路由和带/不带数据位的消息通过AXI地址驱动）。显示了如何通过AXI地址驱动消息代码消息路由和带/不带数据位的消息。

对于供应商定义的消息，供应商定义的消息报头[127：64]通过AXI从属于PCIe地址转换寄存器驱动这将在[*AXI从属于PCIe地址转换寄存器*](#_bookmark73)部分进行解释。

* + 1. AXI从属（错误）响应

应用层可以使用AXI从属响应控制寄存器为多种错误类别和TLP类型配置AXI从属响应代码（AXI xRESP、*axi\_s\_bresp、axi\_s\_rresp*

可使用以下选项：

* + - * + 2'b00 -OK
        + 2'b10 -SLVERR
        + 2 'b11-降低值2' b 01被保留。

[*表8.10*](#_bookmark84)显示了不同的错误类别。

表8.10：AXI从属误差响应类别

|  |  |
| --- | --- |
| 类别 | 描述 |
| 1 | 未成功完成状态。  此类别包括接收到完成状态不成功的完成情况，即*完成中止*（CA）、*未知请求*（UR）、*重复请求状态（RRS）*。  此类别的默认错误代码是SLVERR。 |
| 2 | 完成错误，非成功完成状态。  此类别包括所有完成错误，但不包括未成功的完成状态。例如，完成错误、完成字节计数不匹配等。  此类别的默认错误代码是SLVERR。 |
| 3 | 链接中断事件。  此类别包括在链路断开事件刷新期间由AXI网桥生成的响应此类别的默认错误代码是SLVERR。 |
| 4 | （非）可预取的基础/限制过滤。  此类别包括未通过（非）可预取基本/限制检查并被AXI Bridge过滤的内存事务的响应此类别的默认错误代码是SLVERR。 |
| 5 | AXI Bridge检测到错误。  此类别包括AXI Bridge检测到的错误情况，这些错误情况通常是应用层错误的结果，并使TLP生成变得不可能，例如：使用未映射的业务类、未知的请求、不匹配的区域访问。  此类别的默认错误代码是DECERR。此类别对于所有TLP类型都是常见的。 |

[*表8.11*](#_bookmark85)显示了适用于不同类型TLP的错误类别这反映在AXI从属响应控制寄存器字段中。

 注：5类是所有TLP类型的通用类别。 这是因为，在5类错误的情况下，可能无法确定TLP类型。

表8.11：适用于不同TLP类型的错误类别

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TLP类型 | 类别 | | | |
| 1 | 2 | 3 | 4 |
| Cfg | 是的 | 是的 | 是的 | 没有 |
| IO | 是的 | 是的 | 是的 | 没有 |
| Mem | 是的 | 是的 | 是的 | 是的 |

表8.11：适用于不同TLP类型的错误类别

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TLP类型 | 类别 | | | |
| 1 | 2 | 3 | 4 |
| AtomicOp | 是的 | 是的 | 是的 | 是的 |
| DMWr | 是的 | 是的 | 是的 | 是的 |
| MSG | 没有 | 没有 | 是的 | 没有 |

[*表8.12*](#_bookmark86)显示了AXI Subordinate响应（AXI xUSER、*axi\_s\_buser、axi\_s\_ruser*）的详细错误代码

表8.12：AXI下级响应详细错误代码（xUSER、axi\_s\_buser、axi\_s\_ruser）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 误差来源 | 错误代码（'b） | 错误代码（% d） | 错误说明 | 类别 |
| - | 00000 | 0 | 没有错误。 | - |
| PCIe核 | 00010 | 2 | 完井TLP中毒。 | 2 |
| PCIe核 | 00100 | 4 | 收到供应商ID访问的RRS状态。当设置CRS软件可见性且接收到CFG访问供应商ID地址的RRS Sta- tus时，设置此代码。 | 1 |
| PCIe核 | 00110 | 6 | 请求因完成而终止，但没有数据，或者请求与来自远程端的完成之间存在字节计数不匹配。 | 2 |
| PCIe核 | 01000 | 8 | 收到的完成具有与未完成请求相同的PCIe标签 | 2 |
| PCIe核 | 01010 | 10 | 起始地址错误。在完成头低地址位- er不匹配的下一个预期的字节的请求开始地址。 | 2 |
| PCIe核 | 01100 | 12 | 请求已终止，因为它不在类型1基本/限制范围内。仅适用于RP。 | 4 |
| PCIe核 | 10000 | 16 | 请求通过完成终止。 | 2 |
| PCIe核 | 10010 | 18 | 请求由FLR终止，目标是生成请求的函数。 | 2 |
| PCIe核 | 10100 | 20 | 请求因内存数据包的意外RRS或RCB违规而终止。 | 2 |
| PCIe核 | 11000 | 24 | 请求被UR状态的完成终止（包括规范允许推断UR完成的其他情况）。 | 1 |
| PCIe核 | 11010 | 26 | 请求被具有CA状态的完成终止。 | 1 |

表8.12：AXI下级响应详细错误代码（xUSER、axi\_s\_buser、axi\_s\_ruser）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 误差来源 | 错误代码（'b） | 错误代码（% d） | 错误说明 | 类别 |
| PCIe核 | 11100 | 28 | 请求因具有RRS状态的完成而终止。 | 1 |
| 阿西大桥 | 00011 | 3 | 未知请求-边带描述程序或区域配置错误. | 5 |
| 00101 | 5 | 未映射的流量类。 | 5 |
| 00111 | 7 | 基于区域的访问与任何区域都不匹配。 | 5 |
| 01001 | 9 | （非）可预取基本/限制检查失败。  仅适用于RP。 | 4 |
| 10101 | 21 | AXI网桥链路断开事件刷新。 | 3 |

 注：AXI次级R响应中有误差的数据将设置为全1。当启用配置RRS软件可见性时，此规则有一个例外，即对配置读取请求的RRS状态完成的错误响应。在这种情况下，PCIe规范要求返回的数据为0xFFFF 0001。

* 1. AXI经理

AXI管理器接口结束PCIe控制器的入站路径它也可以与DMA模块共享

它符合*AXI-4规范*并包含AXI-5原子事务功能。AXI管理器模块将入站PCIe事务层数据包（请求）转换为AXI事务（写入或读取），

如果需要，基于AXI响应生成PCIe TLP（完成）。控制器可能有多个AXI管理器接口（取决于客户端接口的数量基于VC将VoIP流量路由到适当的CIF AXI管理器。

* + 1. PCIe到AXI地址转换

SoC地址空间可以与PCIe地址空间完全不同，因此AXI Manager提供了将PCIe BAR映射到AXI地址空间并为传入流量执行地址转换的可能性。

* + - 1. PCIe消息

PCIe消息TLP没有地址。AXI管理器提供两个寄存器（*inb\_at\_msg\_addr0、inb\_at\_msg\_addr1*），可在其中指定这些请求的目标区域来自这些寄存器的数据被组合以形成AXI空间中的目标区域的64位地址。

此地址必须对齐4KB。

* + - 1. RP模式

在RP模式下，有两个PCIe BAR可用。AXI管理器提供三组寄存器-用于BAR 0（*inb\_at\_rp\_bar0\_addr0、inb\_at\_rp\_bar0\_addr1*）、BAR 1（*inb\_at\_rp\_bar1\_addr0、inb\_at\_rp\_bar1\_addr1*）和所谓的“BAR 7”（不匹配）-用于丢失BAR 0或BAR 1的请求（*inb\_at\_rp\_bar7\_addr0、inb\_at\_rp\_bar7\_addr1*）。

来自*addr 0*和*addr 1*寄存器的数据被组合以形成AXI空间中的区域的64位地址

对于以BAR 0和BAR 1为目标的请求，最终的AXI地址构造如下-高位从寄存器中获取低位数取决于给定的BAR孔径。

在不匹配（“BAR 7”）的情况下使用相同的机制-高位从寄存器获取，而低位从TLP地址获取，但低位的数量取决于来自*inb\_at\_rp\_bar7\_addr 0*寄存器的*PASS\_BITS*字段，因为BAR 7形式上不存在，因此其孔径未知。

* + - 1. EP模式

每PF

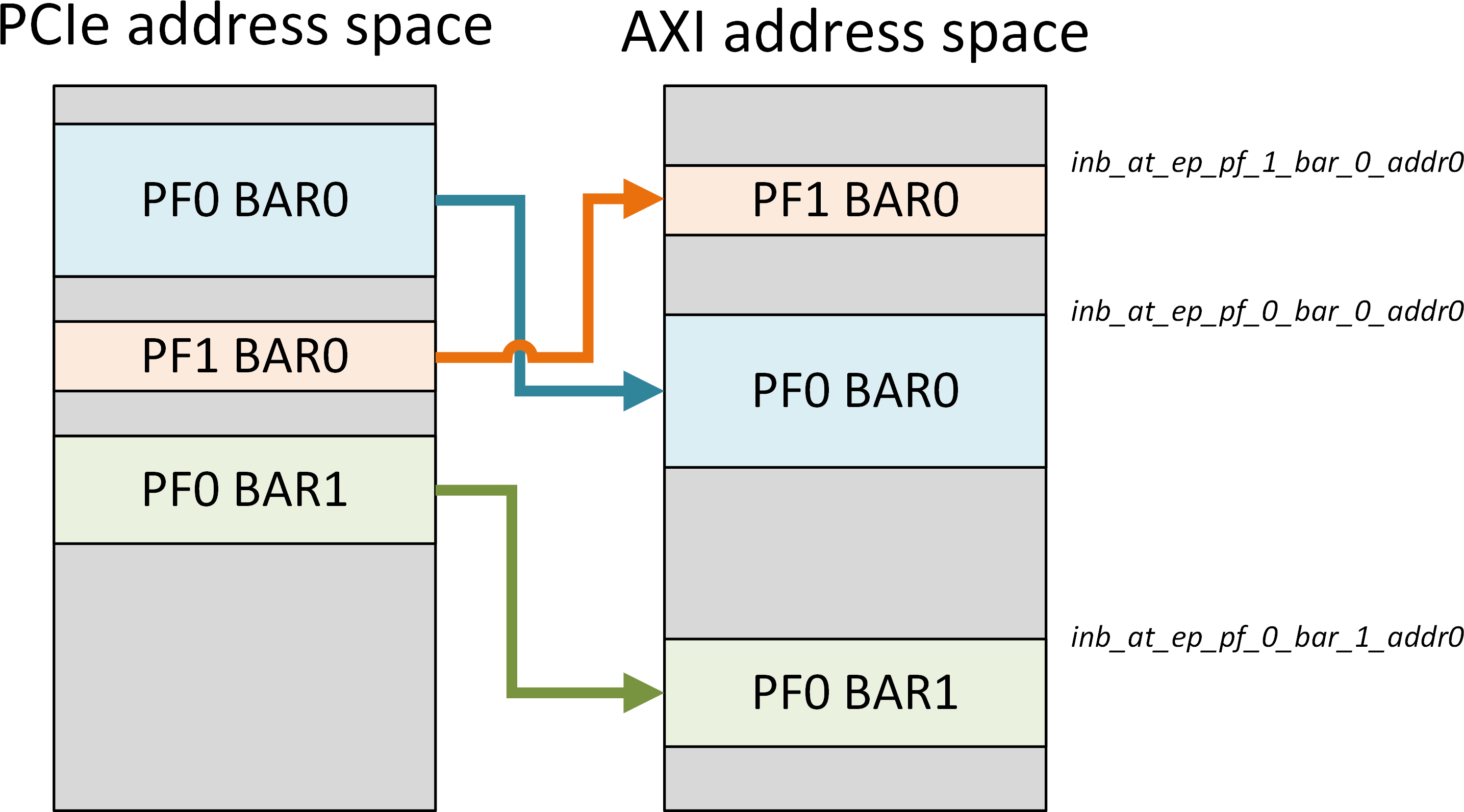
在EP模式下，每个物理功能有7个PCIe BAR：从0到5的BAR和扩展ROM BAR（“BAR 6”）。

AXI管理器为每个PF内的每个BAR提供一组寄存器（*inb\_at\_ep\_pf\_X\_bar\_Y\_addr0*、*inb\_at\_ep\_pf\_X\_bar\_Y\_addr1*）。来自addr 0和addr 1寄存器的数据被组合以形成AXI空间中的区域的64位地址

对于以BAR 0至6为目标的请求，最终AXI地址构造如下：

* + - * + 高位从寄存器
        + 从TLP地址降低。

低位数取决于给定的BAR孔径。未命中所有BAR的TLP地址（所谓的“BAR 7”-无匹配）不会被转换，并按原样传递。



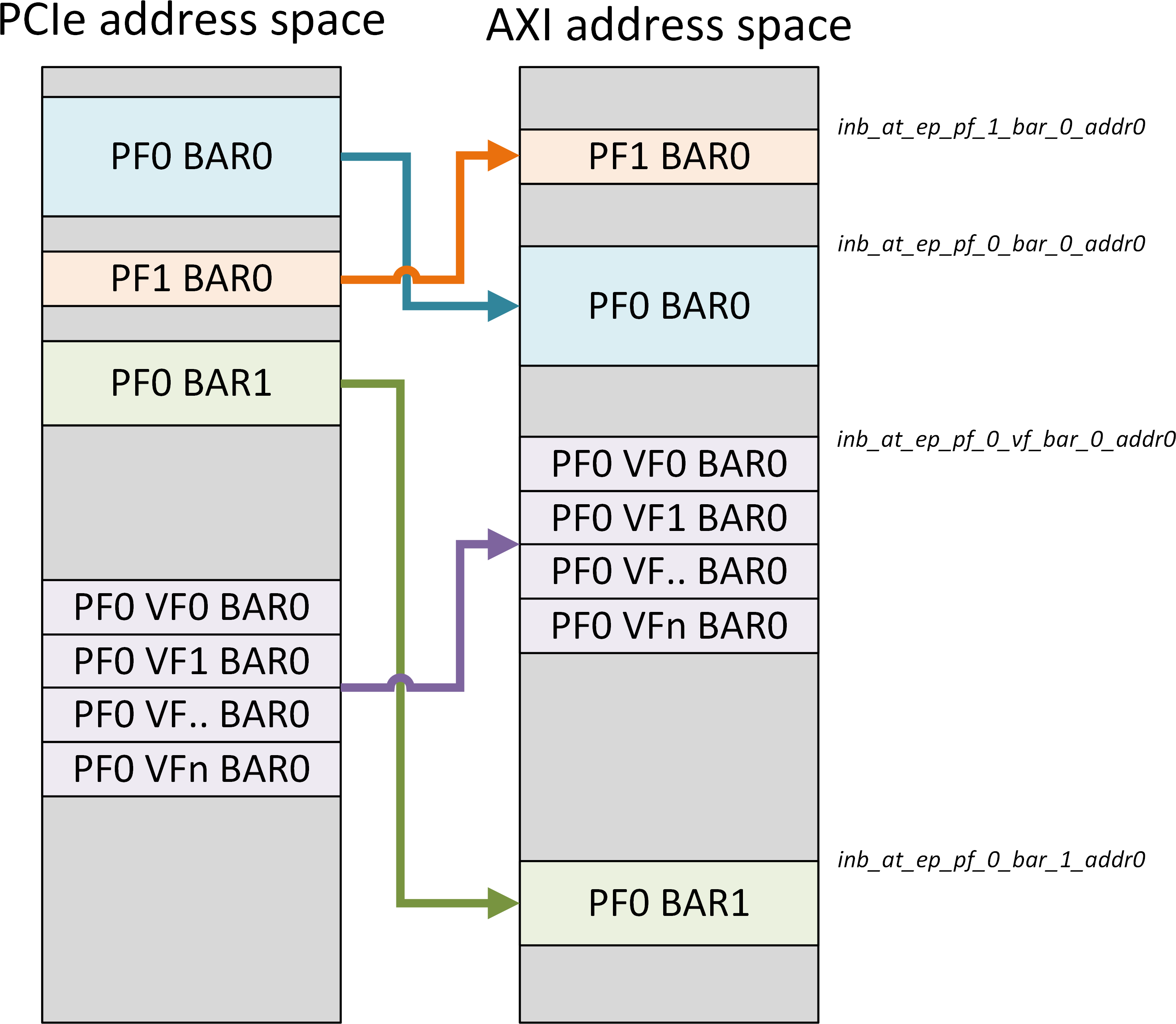
**图8.18：**PF BAR从PCIe到AXI地址空间的映射示例

根据VF

在EP模式中，0到N个虚拟功能可以被分配给物理功能。当PF分配了1到N个VF时，本节适用每个VF有6个PCIe BAR：BAR从0到5。分配给属于给定PF的VF的BAR0的存储器空间是连续的。见[*图8.19*](#_bookmark90)。AXI管理器为每个PF内的每个VF BAR提供一组寄存器（因此为6组-*inb\_at\_ep\_pf\_X\_vf\_bar\_Y\_addr0*、*inb\_at\_ep\_pf\_X\_vf\_bar\_Y\_addr1*）。来自addr 0和addr 1寄存器的数据

被组合以形成AXI空间中的区域的64位地址，即VF 0存储器空间的开始。最终AXI地址的高位取自寄存器，并根据相关VF编号和VF BAR孔径进行调整

低位取自TLP地址。低位数取决于给定的VF BAR孔径。



**图8.19：**PF和VF BAR从PCIe到AXI地址空间的映射

例如，*inb\_at\_ep\_pf\_0\_vf\_bar\_3\_addr0*（VF BAR3）= 0x0480，VF BAR3孔径= 128b（0x80），将PCIe空间中地址为0xFF72的PF 0的VF 2作为目标的TLP转换为：

* + - * + 高位：
        + 低位：0x72（7位）
        + 最终地址= 0x05F2。

CXL

CXL 1.1

EP在CXL 1.1模式下有两个额外的内存区域- RCRB和MEMBAR 0。AXI管理器为这些区域提供两组寄存器（*inb\_at\_ep\_cxl\_rcrb\_addr0、inb\_at\_ep\_cxl\_rcrb\_addr1*和*inb\_at\_ep\_cxl\_membar0\_addr0*、*inb\_ab\_ep\_cxl\_membar0\_addr1*）。

来自*addr 0*和*addr 1*寄存器的数据被组合以形成AXI空间中的区域的64位地址

对于以这两个区域为目标的请求，最终AXI地址构造如下-高位从寄存器获取低位数取决于给定的BAR孔径。

CXL 2.0、3.0

在CXL 2.0及更高版本中，不需要配置这些寄存器，因为在此模式中使用标准PCIe BAR机制

* + 1. AXI管理器写访问

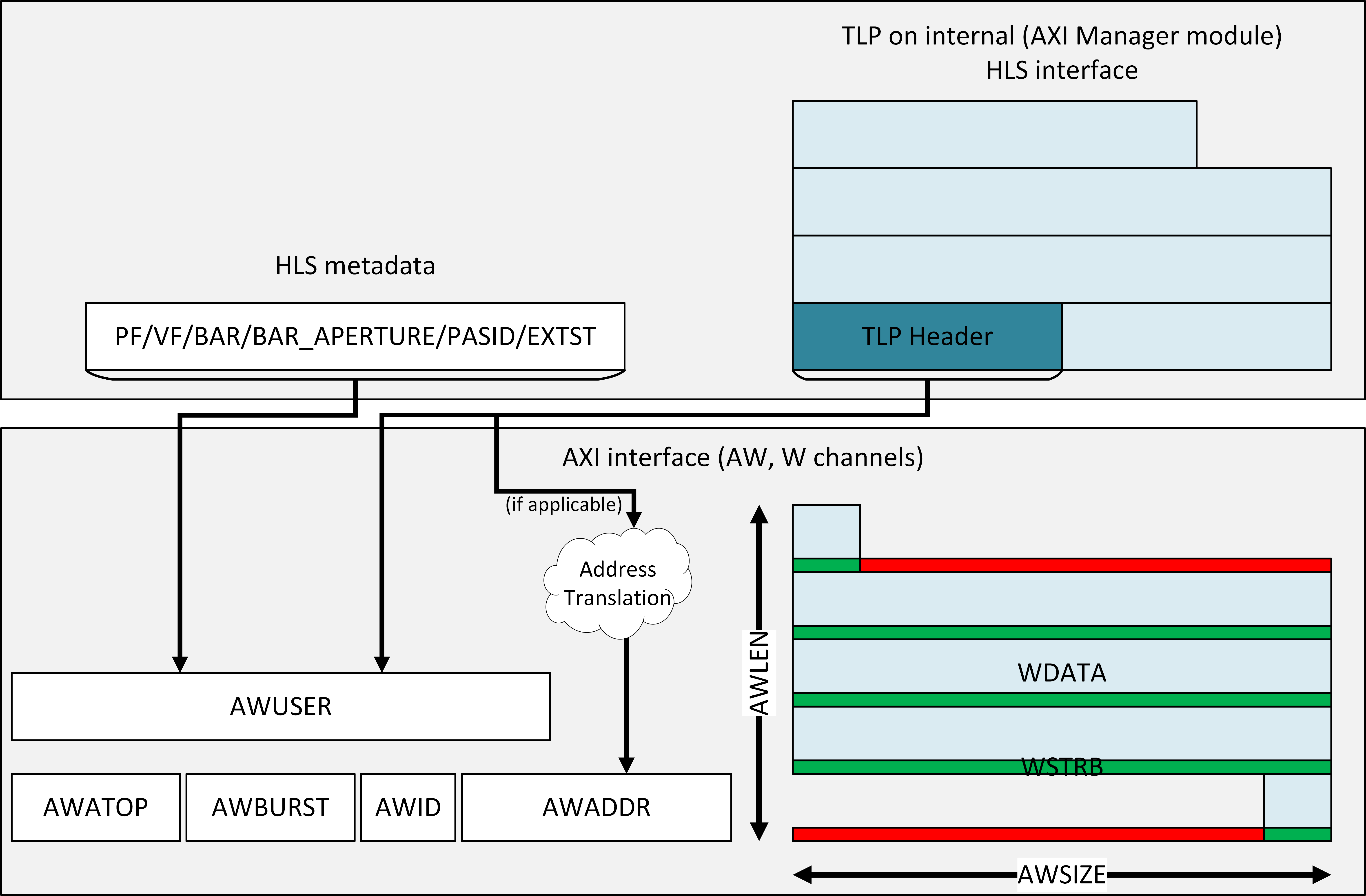
本节介绍PCIe写请求。已发布的内存写入和消息以及未发布的IO写入、可延迟的内存写入和原子操作被转换为AXI管理器写入。

* + - 1. AXI ID池

AXI Manager将AWID池分为几个部分，取决于支持的TLP类型（AtomicOp，Posted，其他Non-Posted）。这些部分的存在及其大小取决于控制器配置。有关AWID池的详细信息可以在*<package\_path>/docs/cdnhpa\_ips\_axi\_m\_axid.html中找到。*

* + - 1. 请求转换

[*图8.20*](#_bookmark93)显示了PCIe写请求TLP如何转换为AXI写事务。



**图8.20：**将PCIe写请求TLP转换为AXI写事务

AXI管理器模块接收序列化TLP。这些中的每一个都伴随有分组元数据向量，其存储与给定分组相关的PCIe信息中的TLP报头被剥离，并且其内容与元数据向量一起被用于形成AW信道传输。

AW送

* + - * + *AWADDR*- 从标头获取TLP地址并执行PCIe到AXI地址转换
        + *AWID*- AXI ID的分配取决于TLP类型和AXI Manager LUT容量（在非发布请求的情况下）有关更多信息，请参见[*AXI ID池*](#_bookmark92)
        + *AWUSER*- TLP报头与其他PCIe信息（目标PF/VF、BAR命中、BAR\_APERTURE、PASID、扩展引导标签等）一起出现在该信号中。- 详细信息可以在*<package\_path>/docs/cdnhpa\_ips\_IO\_Bullection. html*中找到），
        + *AWATOP*- 值6 'b 000000表示非原子访问，非零值表示原子事务。允许的信号在专门讨论AtomicOp请求的章节中描述（原子事务可能会影响*AWLEN*和*AWSIZE*信号，遵循AXI-5原子事务规则），

不需要完全兼容的AXI-5结构，如果AXI-4结构可以遵循AXI-5原子事务规则，

* + - * + *AWLEN*- 数据突发的长度根据TLP有效载荷长度和地址偏移计算（取决于所选的数据路径宽度），
        + *AWSIZE*-导致单拍AXI突发的存储器TLP将具有最小可能*的AWSIZE*;对于IO TLP（总是导致单拍AXI突发）*，AWSIZE*将总是指示4字节;导致多拍AXI突发的存储器TLP将具有*最大*可能的*AWSIZE*-相等的数据路径宽度，

 注意事项：

*AWSIZE*为每个TLP计算一次，当发布请求发生某些条件（在[*发布请求-拆分*](#_bookmark94)部分中描述）并且突发必须*拆分*为多个AXI事务时，所有这些事务都将具有相同的最大可能*AWSIZE*-这可能是次优的对于选定的单拍突发（*拆分*的结果）。

表8.13：512位数据路径宽度的AWLEN和AWSIZE信号示例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TLP参数 | | AXI AW通道 | | | 评论 |
| 地址 | 长度字节 | AWADDR | AWLEN | AWSIZE | - |

表8.13：512位数据路径宽度的AWLEN和AWSIZE信号示例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TLP参数 | | AXI AW通道 | | | 评论 |
| 0x3 | 1 | 0x3 | 0x00 | 0x0 | 单字节。 |
| 0x4 | 4 | 0x4 | 0x00 | 0x2 | 访问的第一个字节与自然地址边界对齐。 |
| 0x10 | 8 | 0x10 | 0x00 | 0x3 |
| 0x2 | 4 | 0x2 | 0x00 | 0x3 | 访问的第一个字节未与自然地址边界 |
| 0xF | 2 | 0xF | 0x00 | 0x5 |
| 0x3F | 2 | 0x3F | 0x01 | 0x6 | 多次心跳。全*AWSIZE*。 |
| 0x20 | 96 | 0x20 | 0x01 | 0x6 |

* + - * + *AWBURST*- 对于非原子突发，始终为INCR

W通道

* + - * + *WDATA-*TLP有效负载在此总线上传递;数据重新对齐，因此第一个字节的位置与目标地址匹配，
        + *WSTRB*- AXI管理器模块为每个脉冲串计算适当的选通，以便仅标记有效字节*WSTRB*允许处理PCIe写TLP的特殊情况：

零长度写入。请参见[*特殊情况零长度写入*](#_bookmark99)。

非连续字节启用。请参见[*特殊情况非连续字节启用*](#_bookmark100)。

* + - 1. 已发布的请求

内存写入和消息是AXI管理器模块支持的发布请求

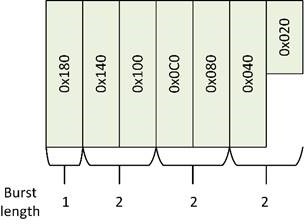
B通道对已发布请求的响应仅用于维护已发布和未发布TLP的内部顺序-忽略BRESP代码。

以下部分介绍已发布请求拆分的条件。

* + - 1. 已发布请求-拆分

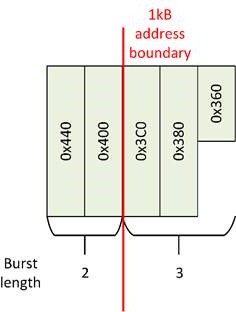
发布的请求的大小可能达到4kB，并且从系统的角度来看，可能希望将大的数据包划分为较小的块。Posted Request*拆分*可能是两种情况的结果，是以下配置变量设置的结果

* + - * + AXI管理器写入突发长度限制（[*图8.21*](#_bookmark95)），



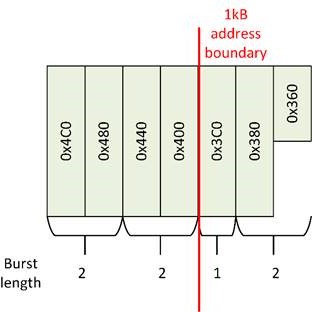
**图8.21：**发布请求拆分-突发长度限制为2

* + - * + 1kB地址边界，如果使能（[*图8.22*](#_bookmark96)）。



**图8.22：**Posted Request拆分-在1 kB地址边界上拆分

可能会发生两个条件同时处于活动状态-在这种情况下，将应用强制最短分割在[*图8.23*](#_bookmark97)中，AXI写入由7个节拍组成。由于将脉冲串长度限制为最多2次搏动，因此进行了第一次分割，但对于第二次分割，两个条件均适用-1 kB地址边界规则强制1次搏动的较短脉冲串长度



**图8.23：**已发布请求拆分-两种情况同时发生

* + - 1. 未发布的写入请求

IO写入、可延迟内存写入和原子操作是AXI管理器模块支持的非发布写入请求（后两种类型取决于相应的配置变量）。

有关其他信息，请参见[*原子操作*](#_bookmark114)

IOWr和DefMemWr需要PCIe完成。当收到AXI事务的B通道响应时，它由AXI管理器模块生成，其完成状态字段值取决于BRESP代码和BUSER值（用于DefMemWr），如[*表8.14*](#_bookmark98)所示。

表8.14：IO写入和可延迟内存写入的完成状态的可能值

|  |  |  |  |
| --- | --- | --- | --- |
|  | IOWr | DefMemWr | |
| BUSER -位0 | |
| 0 -接受 | 1 -延期 |

表8.14：IO写入和可延迟内存写入的完成状态的可能值

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| BRESP | 好 | SC | SC | RRS |
| SLVERR/DECERR | CA | CA | CA |

 注意：未过帐的写请求永远*不会*拆分为多个AXI事务。

* + - 1. 未完成请求

AXI Manager模块生成的未完成AXI Manager写入事务（已发布和未发布写入请求）的数量取决于所选配置（从4到4096）。当未完成事务计数器饱和时，AXI*READY*和*VALID*信号将被屏蔽每当收到B响应（原子事务的B和R响应）时，未完成事务计数器将

每个拆分请求都计入未完成交易限额。

* + - 1. 错误处理

已发布请求的BRESP错误将被忽略。

非发布写入请求的BRESP错误会影响其完成状态。见[*表8.14*](#_bookmark98)。

* + - 1. 特殊情况零长度写入

1 DW的内存写入请求和未启用的字节称为零长度写入。此类请求转换为单字节AXI写入（AWLEN = 0x00，AWSIZE = 0x0），所有写入选通均禁用。

* + - 1. 特殊情况非连续字节启用

*PCIe规范*允许具有非连续字节使能的存储器写入请求用于具有与8字节边界（QW-四字）对齐的地址的1个DW TLP和2个DW TLP对于此类请求，将生成适当的写入选通，因此只有启用的字节在AXI上标记为有效。

* + 1. AXI管理器读取访问

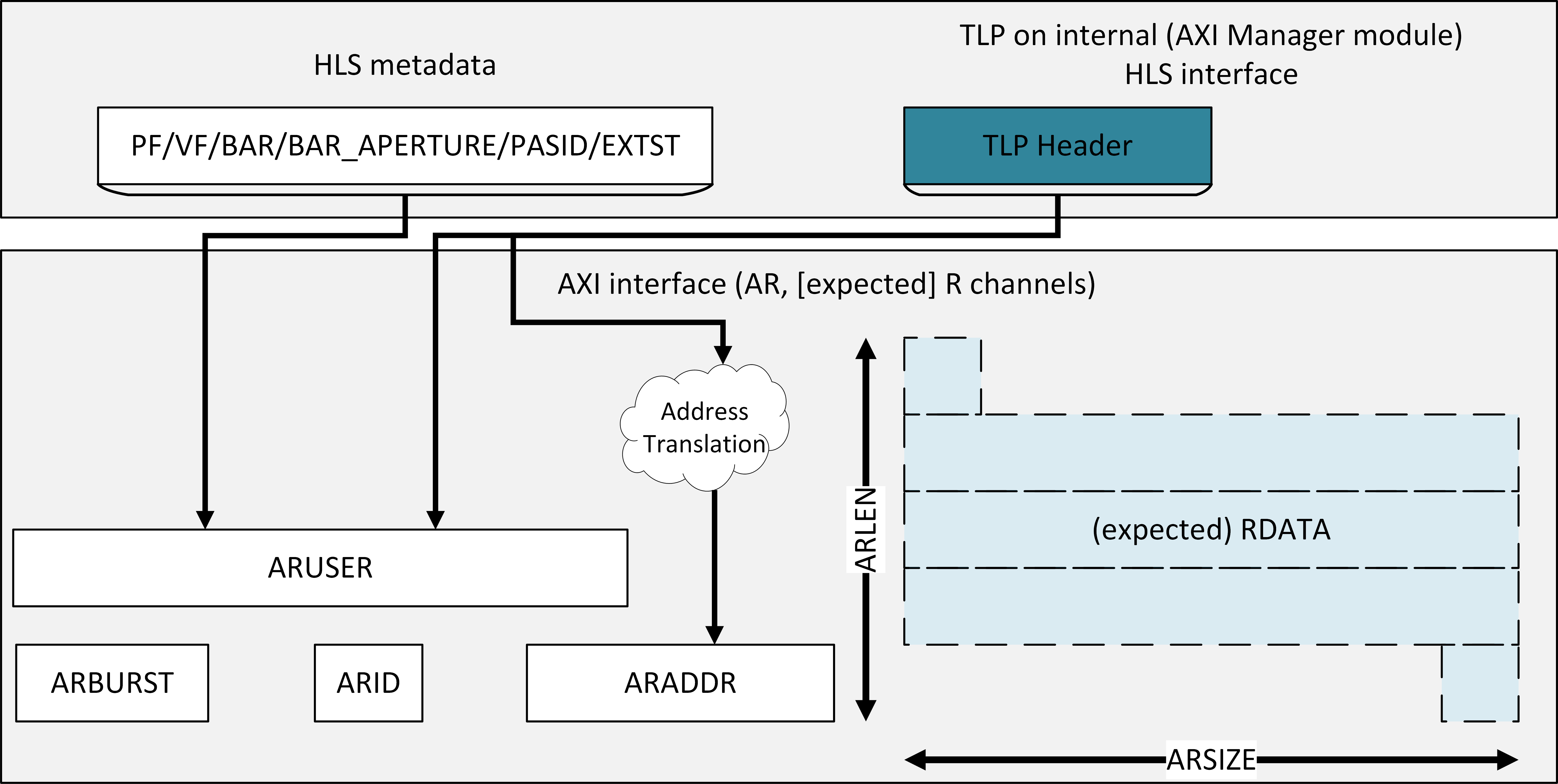
本节介绍PCIe读取请求（IO读取和内存读取）如何转换为AXI管理器读取。

* + - 1. AXI ID池

AXI Manager将ARID池分为几个部分，取决于支持的TLP类型（AtomicOp，其他非发布）。这些部分的存在及其大小取决于控制器配置。有关ARID池的详细信息可在*<package\_path>/docs/cdnhpa\_ips\_axi\_m\_axid.html中找到。*

* + - 1. 请求转换

[*图8.24*](#_bookmark103)显示了PCIe读取请求TLP如何转换为AXI读取事务。



**图8.24：**PCIe读请求TLP到AXI Manager读事务的转换

AXI管理器模块接收序列化TLP。这些中的每一个都伴随有分组元数据向量，其存储与给定分组相关的PCIe信息中的一些。TLP报头与元数据向量一起用于形成AR信道传输。

AR通道

* + - * + ARADDR-TLP地址从报头中取出，并执行PCIe到AXI地址转换
        + *ARID*- 根据TLP类型和AXI Manager LUT容量分配AXI ID请参阅[*AXI ID池*](#_bookmark102)了解更多信息。
        + *ARUSER*- TLP报头与其他PCIe信息（目标PF/VF、BAR命中、BAR\_APERTURE、PASID、扩展引导标签等）一起出现在该信号中。- 详细信息可以在*<package\_path>/docs/cdnhpa\_ips\_IO\_Bullection. html*中找到），
        + *ARLEN*- 根据TLP长度和地址偏移计算数据突发的长度（取决于所选数据路径宽度），
        + *ARSIZE*- 导致单拍AXI突发的存储器TLP将具有最小可能*的ARSIZE*;对于IO TLP（总是导致单拍AXI突发）*，ARSIZE*将总是指示4字节;导致多拍AXI突发的存储器TLP将具有*最大*可能的*ARSIZE*-相等的数据路径宽度，

**注：***ARSIZE*在每个TLP中计算一次，当内存读取请求出现[*内存读取请求拆分*](#_bookmark104)中所述的某些条件时，突发必须*拆分*为多个AXI事务，所有这些事务都将具有相同的最大可能*ARSIZE*-这对于选定的单拍突发（*拆分*结果）可能是次优的。

表8.15：512位数据路径宽度的ARLEN和ARSIZE信号示例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 案件编号 | TLP参数 | | AXI AW通道 | | | 评论 |
| - | 地址 | 长度字节 | ARADDR | Arlen | ARSIZE | - |
| 1 | 0x3 | 1 | 0x3 | 0x00 | 0x0 | 单字节。 |
| 2 | 0x4 | 4 | 0x4 | 0x00 | 0x2 | 访问的第一个字节与自然地址边界对齐。 |
| 3 | 0x10 | 8 | 0x10 | 0x00 | 0x3 |
| 4 | 0x2 | 4 | 0x2 | 0x00 | 0x3 | 未与自然地址边界 |
| 5 | 0xF | 2 | 0xF | 0x00 | 0x5 |
| 6 | 0x3F | 2 | 0x3F | 0x01 | 0x6 | 多次心跳。完整*的ARSIZE*。 |
| 7 | 0x20 | 96 | 0x20 | 0x01 | 0x6 |

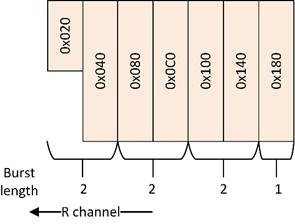


**图8.25：**512位数据路径宽度的ARLEN和ARSIZE信号示例

* + - * + *ARBURST*-始终INCR。
      1. 内存读取请求拆分

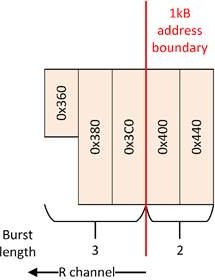
存储器读取请求的大小可能达到4kB，并且从系统的角度来看，将大的访问划分成较小的块可能是期望的在以下情况下可能会发生这种情况：

* + - * + 达到猝发内最大搏动数的限制*。* 见[*图8.26*](#_bookmark105)。
        + 最大有效负载大小设置，限制PCIe完成的大小例如，128字节，见[*图8.26*](#_bookmark105)。[*图8.26*](#_bookmark105)



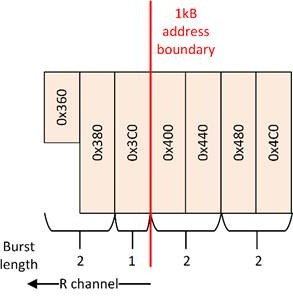
**图8.26：**内存读取请求拆分-突发长度限制为2拍（使用选定配置或128字节MPS）

* + - * + 如果启用，则跨越1kB地址边界见[*图8.27*](#_bookmark106)。



**图8.27：**内存读取请求拆分-在1 kB地址边界上拆分

可能会发生多个条件同时处于活动状态-在这种情况下，将应用强制最短拆分规则。在[*图8.28*](#_bookmark107)中，AXI读取由7个节拍组成由于将脉冲串长度限制为最大2次搏动，因此进行了第一次分割



**图8.28：**内存读取请求拆分-同时发生两种情况

当拆分AXI管理器模块确保满足读取完成边界规则时，可能的最长突发仍可能缩短一点以强制执行RCB合规性。

* + - 1. AXI响应转换

来自R通道的每个响应都会转换为读取请求的PCIe完成AXI管理器模块存储请求上下文，例如，满足请求的剩余字节计数。

所有R响应搏动的RRESP代码必须为OK，因此生成带有数据和SC完成状态的完成当至少有一个心跳标记为错误（SLVERR或DECERR）时，将生成具有CA完成状态的完成，并终止请求。

* + - 1. 未完成请求

由AXI Manager模块生成的未完成AXI Manager读取事务（非发布读取请求）的数量可以由所选配置限制（从4到4096）。当未完成事务计数器饱和时，AXI*READY*和*VALID*信号将被每当收到R响应时，未完成交易计数器将递减。

每个拆分请求都计入未完成交易限额。

* + - 1. 特殊情况零长度读取

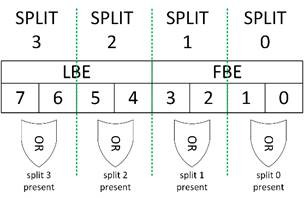
没有启用字节的1 DW的内存读取请求称为零长度读取。这种请求被转换为单字节的AXI管理器读取（ARLEN = 0x00，ARSIZE = 0x0）。具有字节计数等于1的PCIe完成（遵循*PCIe规范*）从针对该访问的R响应生成。

* + - 1. 特殊情况非连续字节启用

*PCIe规范*允许具有非连续字节使能的存储器读取请求用于具有与8字节边界（QW-四字）对齐的地址的针对可预取和不可预取内存空间的此类请求将*默认*以相同的方式处理-相应AXI管理器读取访问的*ARSIZE*将根据部分中的规则计算，并读取目标位置的**所有**

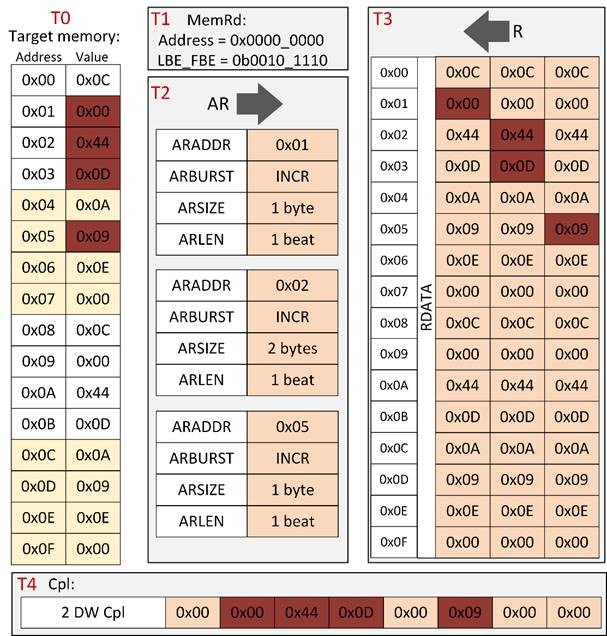
不可预取拆分

不可预取的读取拆分支持可以在IP配置期间启用。支持时，具有非连续字节的内存读取请求可使目标不可预取内存空间拆分为多个1或2字节的AXI管理器读取，以确保仅读取预期字节。



**图8.29：**NPRF拆分规则

[*图8.29*](#_bookmark108)显示了NPRF分割规则（最大值）8字节的内存读请求被分成4个2字节的块。如果每个组至少启用一个字节，则将出现拆分AXI Read-*ARSIZE*取决于启用的字节数（0x 0或0x 1）。



**图8.30：**禁用某些字节的2 DW MemRd TLP的NPRF拆分示例

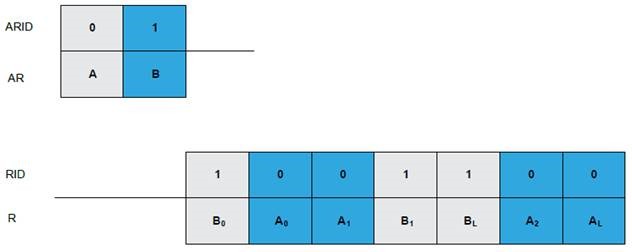
[*图8.30*](#_bookmark109)和[*表8.16*](#_bookmark110)显示了不可预取拆分的示例

表8.16：NPRF划分示例

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 没有 | 地址 | | 分割0 | | 分流1 | | 分裂2 | | 分流3 | |
| LBE | FBE | 地址 | 大小 | 地址 | 大小 | 地址 | 大小 | 地址 | 大小 |
| 1 | 0x4BE4 | | 0x4BE4 | 1b | 0x4BE7 | 1b | - | - | - | - |
| 0000 | 1001 |
| 2 | 0x7738 | | - | - | 0x773A | 2b | 0x773C | 2b | - | - |
| 0011 | 1100 |
| 3 | 0xABCC | | 0xABCD | 1b | 0xABCE | 2b | 0xABD0 | 1b | 0xABD3 | 1b |
| 1001 | 1110 |
| 4 | 0x1234 | | 0x1234 | 1b | - | - | 0x1238 | 2b | 0x123A | 1b |
| 0111 | 0001 |
| 5 | 0x0000 | | 0x0001 | 1b | 0x0002 | 1b | - | - | 0x0007 | 1b |
| 1000 | 0110 |

* + - 1. 去交错缓冲器

如果系统允许交错读取突发数据（参见[*图8.31*](#_bookmark111)），则在生成PCIe完成之前，必须按请求（AXI ID）对其进行排序。AXI管理器解交织缓冲器是AXI管理器模块的一部分，提供R响应解交织功能-每个R响应的所有节拍都存储在解交织缓冲器*仓*中，只有在接收*到RLAST*时才释放突发



**图8.31：**交错R脉冲串

当需要解交织能力时，解交织缓冲器仓的数量应当等于AXI管理器NP RD LUT深度。

bin的大小取决于R突发的最大长度这是三个条件之一的结果

* + - * + AXI管理器读取突发长度限制，
        + 最大支持的最大有效负载大小值，
        + 1kB地址边界分割是否有效。请参见[*内存读取请求拆分*](#_bookmark104)。

如果启用了AtomicOp支持，则AXI管理器解交织缓冲区具有用于原子事务R响应的单独的AtomicOpFIFO

* 1. AXI链路故障处理

链路断开是在链路接通之后的任何时间引起的，也就是说，当LTSSM处于L0并且稍后LTSSM从L0状态转换到DETECT状态时。这通常是一种罕见的事件，在正常操作中不会发生它表示为

*link\_down\_handling\_in\_progress*端口和映射到AXI从属寄存器空间的链路中断指示位

此事件需要在应用程序层通过执行纠正操作（如刷新所有事务和重新启动DMA）来处理

AXI下属

当链路中断事件发生时，AXI下属丢弃所有请求并充分响应，直到应用层重置AXI下属寄存器空间中映射的链路中断指示位只有当所有未完成的AXI交易完成时，链路中断处理才会结束这由*axi\_s\_link\_down\_reset\_flush\_done*指示。

注意事项：

* 如果应用层在*axi\_s\_link\_down\_reset\_flush\_done*为高电平时未重置链路中断指示位并发出新请求，则*axi\_s\_link\_down\_reset\_flush\_done*将被解除断言，以指示链路中断处理重新启动。
* 如果应用层在链路中断处理完成之前重置链路中断指示位，则AXI下属将取消AW、W、AR就绪输出，以停止任何新的AXI请求。

AXI经理

当链路中断事件发生时，AXI管理器等待，直到所有AXI事务都完成并有B或R响应。应该生成的所有PCIe完成都被丢弃。只有当所有未完成的AXI交易完成时，链路中断处理才会结束这由*axi\_m\_link\_down\_reset\_flush\_done*指示。

DMA

DMA与链路断开无关，因此当链路断开事件发生时，它将继续发送预定请求。在这种情况下，DMA发出的所有请求都将因AXI Ext Subordinate错误而失败。如果在控制器IP中使用DMA，则应用逻辑应等待，直到所有DMA通道完成，然后才重置映射在AXI从属寄存器空间中的链路断开指示位

* 1. 原子操作

本节简要描述PCIe原子操作和AXI-5原子事务的概念有几个AXI-5原子事务的例子，让读者更好地理解接口的功能

* + 1. PCIe原子操作

原子操作是单个PCIe事务，其读取目标位置的值，可能写回新值并返回原始值。*读取-修改-写入*序列以原子方式执行。

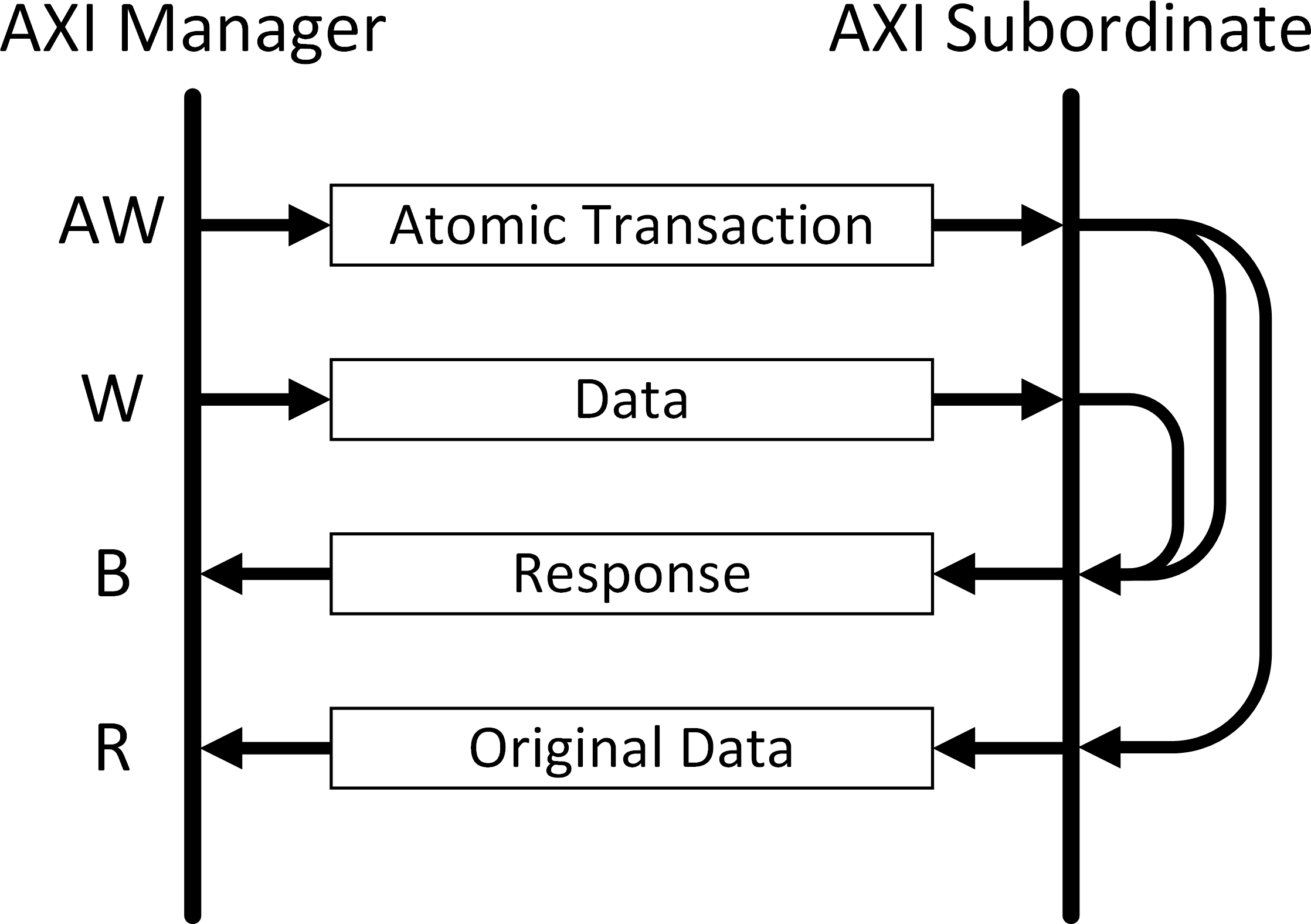
表8.17：PCIe原子操作的类型

|  |  |  |  |
| --- | --- | --- | --- |
| 操作 | 描述 | 操作数 | 可能的操作数大小 |
| Fetch and Add（FetchAdd） | 请求包含单个操作数--*加法*值。读取目标位置，*添加*值，并将修改后的值写回目标位置.返回目标位置 | 1 | 32位  64位 |
| 无条件交换（Swap） | 请求包含单个操作数-*交换*值。读取Tar- get位置，写回*交换*值并返回原始值。 | 1 | 32位  64位 |
| 比较和交换（CAS） | 请求包含两个操作数  - *比较*值和*交换*值。读取目标位置，并将其值与*比较*值进行比较。如果它们相等，*则交换*值被写回到目标位置。如果它们不相等，什么也不会发生。返回目标位置 | 2 | 32位  64位  128位 |

CAS请求包含两个操作数：

* 数据区中的一个总是比较值。
* 一个是互换价值。
  + 1. AXI-5原子交易

原子事务使用AW AXI通道发出，并带有附加*AWATOP*信号，以区分非原子事务和原子事务（数据在W上传输）。它必须使用为AtomicOps保留的AXI ID，这对于W和R通道是常见的。对于原子事务，B（写响应）和R（从目标位置读取原始数据的响应）都是预期的响应（[*图8.32*](#_bookmark118)）。



**图8.32：**原子事务结构

AXI-5规范定义了许多可能具有各种数据操作的原子事务类型，但PCIe仅支持3种原子操作（[*表8.18*](#_bookmark119)）。

表8.18：PCIe AtomicOp映射到AXI-5原子事务

|  |  |  |  |
| --- | --- | --- | --- |
| PCIe AtomicOp类型 | | AXI-5原子事务类型 | AWATOP[5：0]（1），第E2.1.6节 |
| 非原子操作 | 正常请求 | 正常请求 | 0b000000 |
| 原子操作 | FetchAdd | AtomicLoad（ADD操作） | 0b10e000  (bit*e*表示数据的端号） |
| 交换 | AtomicSwap | 0b110000 |
| CAS | 原子比较 | 0b110001 |

上表显示了PCIe原子操作和AXI-5原子事务之间的映射，并包含有效的AWATOP值。不支持所有其他值（其他AXI-5原子事务类型）。

 注：AXI编译器模块生成的AXI AtomicLoads的位e的值可以设置为

*axi\_m\_atomicop\_endianness*配置变量。

* + - 1. 示例

以下列表包含AXI-5原子事务的示例你必须记住：

* + - * + AW（请求）和W（请求数据）之间的相对定时未指定，
        + W和R事务处理的顺序未指定为：

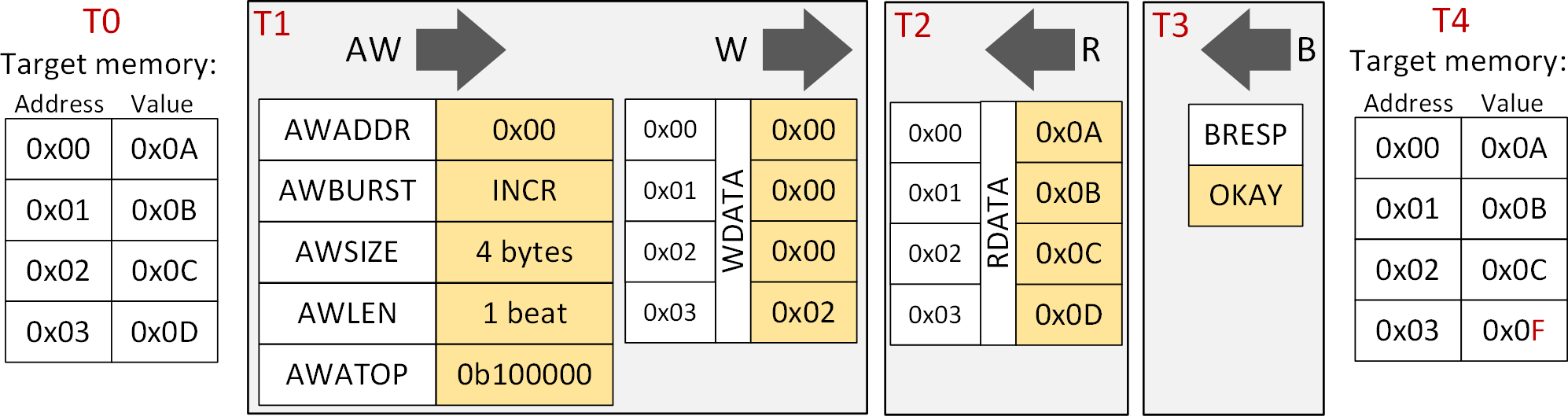
允许下级在发送读数据之前等待所有写数据

允许下级在接受任何写数据之前发送所有读数据

* + - * + 因此，也没有指定B和R响应的顺序（B必须在接收到所有W数据并且原子事务的结果可观察后才给出

以下示例仅旨在阐明AXI-5原子事务的概念，并且仅是可能情况的子集

AXI-5带ADD操作的

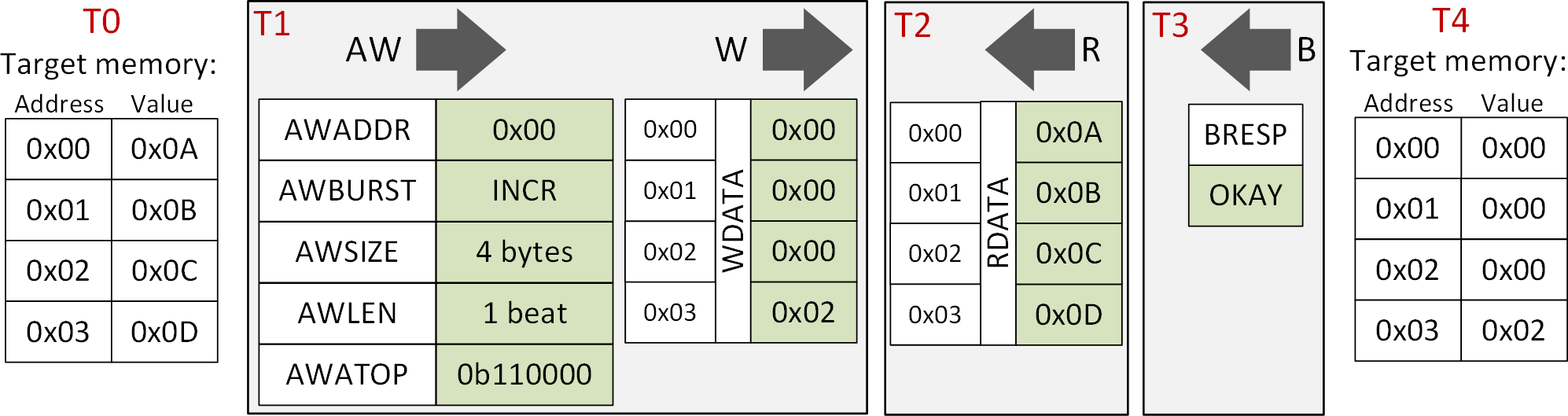


**图8.33：**AXI-5带ADD操作的原子负载

在T0时，地址0x00处的存储器位置具有值0x0D0C0B0A。原子事务在时间T1在AW通道上发出原子事务操作数为0x02000000（W通道）。在时间T2，原始数据返回到R通道上-该值从目标存储器位置读取。在T3处，在B通道上接收到写入响应-原子事务的结果是可观察的。目标内存位置的新值为0x0F0C0B0A

- 将操作数添加到原始值。

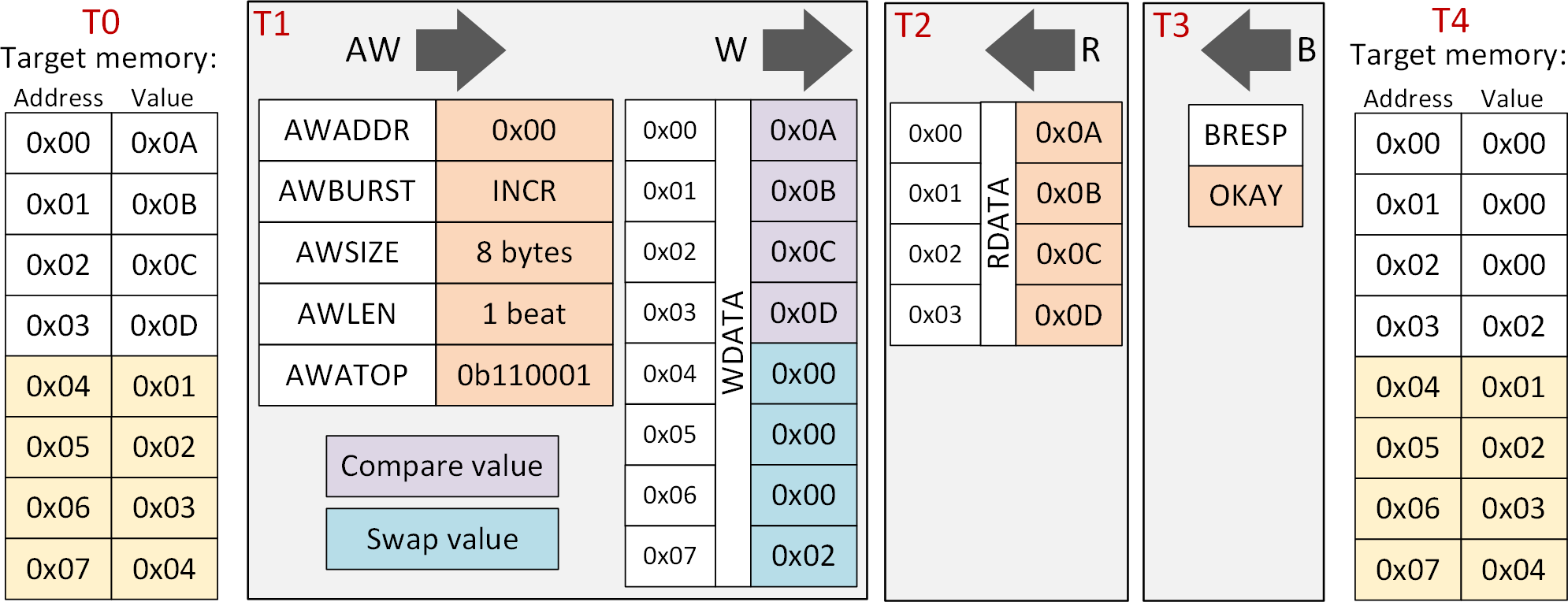
AXI-5原子交换



**图8.34：**AXI-5原子交换操作

原子交换（AWATOP = 6'b110000）是与原子加载类似的事务，但是它在目标存储器位置上执行任何操作，而是将其操作数写入其中。原始数据在R通道上返回，并且在结束时（T4）目标存储器位置值等于原子交换操作数（W通道）。

AXI-5 Atomic与对齐地址



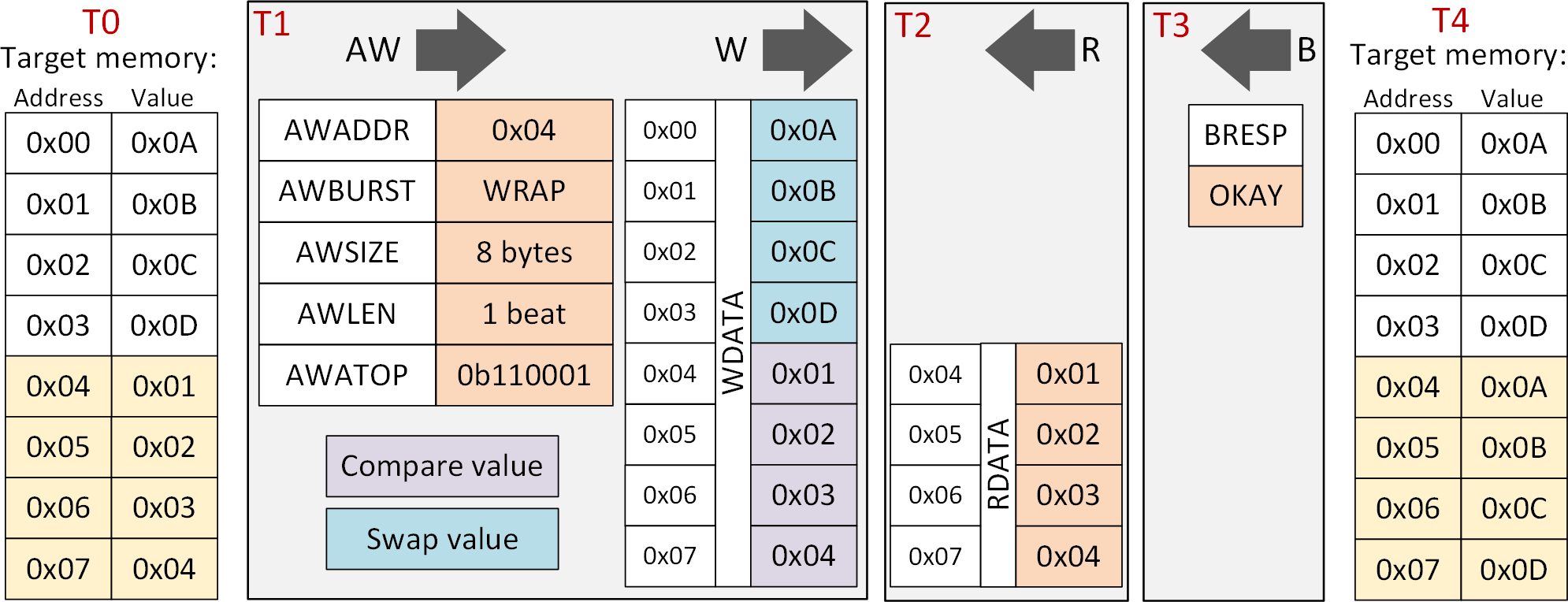
**图8.35：**AXI-5原子比较操作，地址与总突发大小对齐

在T1，AtomicCompare事务在AW通道上发出（AWATOP = 6'b110001）。操作数大小是总事务大小的一半。首先发送比较值，因为事务地址与突发的总大小对齐在T2原始

数据在R通道上返回-入站数据仅为出站数据大小的一半（必须发送前两个操作数

如果目标内存位置等于比较值，它将被交换值覆盖（如上图所示如果这两个值不同，则不会发生任何事情。

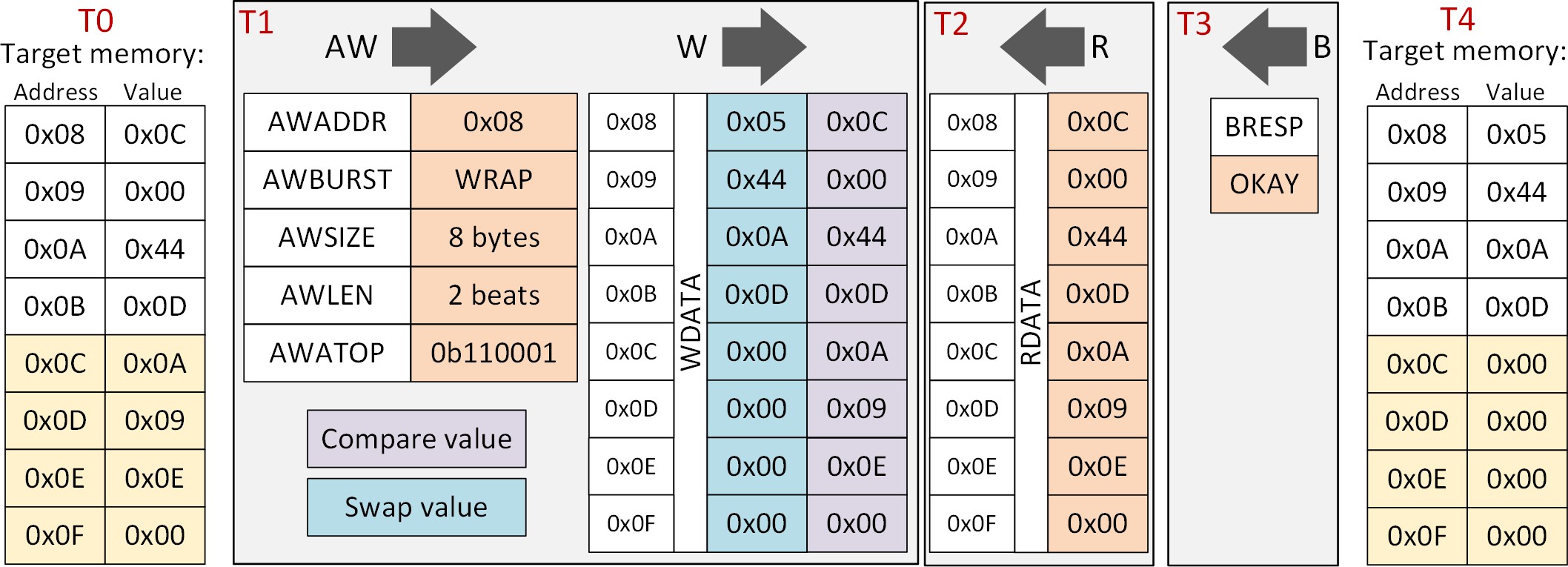
AXI-5原子与未对齐地址比较，AWLEN = 0



**图8.36：**地址未与总突发大小对齐的原子比较操作

在这种情况下，交换值首先发送，因为事务地址与总突发大小不（0x04与8字节）。这在AW请求中通过AWBURST信号等于WRAP来标记。

AXI-5 Atomic与未对齐地址比较，AWLEN> 0



**图8.37：**地址与总突发大小不对齐的原子比较操作（AWLEN> 0）

在这种情况下，突发的总大小是16字节（2拍 \* 8字节）。操作数大小为8个字节。事务地址未与总突发大小对齐，因此交换值在比较值之前发送。比较值等于目标内存位置，因此交换值被写入内存。

* + 1. AtomicOp on AXI Subordinate

使用AXI Subordinate发布AtomicOp有两种可能的方法

* 通过AXI区域

此过程在“PCIe AXI通过区域的从属访问”部分中描述在该方法中，PCIe相关信息存储在AXI区域寄存器中。区域应编程为内存区域。用户必须另外将*axi\_s\_awatop*信号设置为适当的值。

* 使用边带描述符的

该过程在[*AXI通过边带描述符的次级访问部分中*](#_bookmark76)描述。在该方法中，PCIe相关信息通过AWUSER信号给出用户必须另外将*axi\_s\_awatop*信号设置为适当的值。见[*表8.18*](#_bookmark119)。

用户可能会意外错过AtomicOp访问的所有编程区域如果区域选择逻辑没有找到匹配，则在B和R通道上返回适当的错误响应。

 注意事项：

* + *axi\_s\_awsize*和*axi\_s\_awlen*必须匹配AtomicOp TLP有效负载的正确大小
  + 由于在AtomicOp TLP中保留了First BE和Last BE字段，因此W通道的写入选通对于AtomicOp访问没有特定意义要求为AtomicOp访问的有效字节设置写选通，并且写选通是连续的。

为了生成正确的比较和交换AtomicOp请求，用户必须遵循AXI-5规范中关于AXI AtomicCompare原子事务的所有规则。对于地址与突发总大小对齐的访问，*axi\_s\_awburst*必须设置为INCR，“比较”操作数必须放在*axi\_s\_wdata*的第一位，“交换”操作数必须放在第二位。对于地址与突发总大小不对齐的访问，*axi\_s\_awburst*必须设置为WRAP，“swap”操作数必须放在*axi\_s\_wdata*的第一位，“compare”操作数必须放在第二位。

* + 1. AtomicOp on AXI Manager
       1. AtomicOp请求

原子操作TLP由AXI管理器模块以与其他非发布写入请求类似的方式处理（即，报头和PCIe上下文呈现在AWUSER总线上，或者TLP地址经历PCIe到AXI地址转换）。有关其他信息，请参阅[*AXI Manager*](#_bookmark88)，但以符合AXI-5的方式在接口上显示，并[*在AXI-5 Atomic Transactions*](#_bookmark117)中以示例进行了描述，其中AXI Manager模块充当总线管理器。

AXI ID的数量（所有通道通用）仅为原子事务保留，并取决于AXI管理器AtomicOp LUT深度（axi\_m\_atomicop\_lut\_depth配置变量）-AXI管理器模块可能会发出最多如此多的原子事务。

由AXI管理器模块发出的具有ADD操作的AXI 5 AtomicLoad的字节序（PCIe Fetch和Add请求的结果）可以用axi\_m\_atomicop\_endianness配置变量来设置。

在PCIe比较和交换请求具有与总请求大小不对齐的地址的情况下，AXI管理器模块负责改变“比较”和“交换”操作数的顺序以符合AXI-5原子事务规范。

* + - 1. AtomicOp完成

由于原子操作是非发布的，因此AXI Manager模块必须为每个请求生成完成

AXI管理器模块仅在接收到B和R AXI响应时才发送AtomicOp完成两个响应的xRESP代码都必须为OK，因此完成状态为SC。如果任何xRESP代码为SLVERR或DECERR，则将生成带有CA完成状态的完成。

* 1. 信用可用性接口
     1. 概述

本节介绍客户端信用可用性接口逻辑。它由以下已发布和未发布的端口组组成有关IO的更多信息，请参阅IP包中提供的信号列表和说明

* 信用限额/非/张贴标题（缩放）-12位每VC
* 信用限额/非/张贴的有效载荷（缩放）-每VC 16位
* 信用可用于/非/已过帐标头-每个VC 1位
* 信用可用于/非/过帐的有效负载-每个VC 1位
* 对于/非/发布的报头和有效载荷，信用无限-每个VC 2位
  + [0]- 标题学分是无限的
  + [1]- 有效载荷信用是无限的



AXI Credit I/F

crd\_consumed\_header\_per\_VC

crd\_lmt\_/non/posted\_header [VC\*12-1：0]

–

AXI

crd\_lmt\_/non/posted\_payload [VC\*16-1：0]

**=**

crd\_consumed\_payload\_per\_VC

crd\_avail\_/non/posted\_header [VC-1：0]

–

客户端逻辑

=

crd\_avail\_/non/posted\_payload [VC-1：0]

crd\_limit\_header\_per\_VC

X

crd\_lmt\_/non/posted\_inf [VC\*2-1：0]

crd\_limit\_payload\_per\_VC

PCIe

X

保留

[25：16] 第一次见面 有效载荷[14：0]

**图8.38：**信贷可用性

crd极限标度和控制

信用门槛登记（AXI下属）

所有端口都包括每个VC的信用值，这些信用值被打包在向量中，VC0位于LSB，而VCmax位于MSB，如下面的示例所示：

crd\_lmt\_posted\_header[VC\*12-1：0]=

{crd\_lmt\_posted\_header\_VCmax[11：0]，.，crd\_lmt\_posted\_header\_VC1[11：0]，crd\_lmt\_posted\_header\_VC0[11：0]}

客户端逻辑处的信用限制报头和有效载荷值等于PCIe信用限制减去由AXI逻辑消耗的信用有时，当AXI逻辑消耗比当前PCIe信用限制更多的信用时，信用限制值可以变成编码为2的补码的负数。在这种情况下，将停止分组传输，直到从PCIe远程设备接收到新的信用。

* + 1. 信用评级

客户端逻辑处的所有信用值基于PCIe规范中定义的信用缩放规则来缩放。12-比特报头信用限制端口宽度和16比特有效载荷信用限制端口宽度可以保持具有对应于比例因子4的比例乘数16的值

* + 1. 信用阈值寄存器

映射在AXI次级寄存器空间中的信用阈值寄存器保存信用值，该信用值与当前信用限制进行比较阈值以4DW单位给出。

* + 1. 链路断开和低功耗模式

在PCIe链路断开期间，所有信用都被设置为不定式，并且所有可用输出都是活动的。在PCIe链路断开退出事件时，信用值被重置，并且必须从PCIe远程设备授予新信用如果信用限制改变或信用被消耗，则在PCIe低功率模式期间更新信用I/F输出

* + 1. 基于DMA信用的流

如果DMA可用，则它通过作为控制器IP的一部分的AXI从属接口传输请求。DMA请求基于信用可用性进行门控，以防止在信用不存在时阻塞AXI从属请求管道

# 第9章：其他接口

第9章：其他接口

* 1. 中断接口

当IP被选择为EP模式时，应用逻辑可以使用该接口来生成由INTx、MSI和MSI-X组成的中断

* + 1. INTx或旧版以太网接口

[*图9.1*](#_bookmark131)显示了SSL断言和解除断言消息。 注意事项：

以下时序图仅供参考和示例。



时钟

Intx\_in[0]

int\_ack[0]

intx\_disable

**图9.1：**INTx接口

上面的时序图是针对INTA显示的，它使用intx\_in[0]和int\_ack[0]。如果支持，则INTB、C和D分别由intx\_in和int\_ack比特1、2、3表示

* + 1. MSI-X中断

MSI-X接口用于客户端发送MSIX中断，如[*图9.2*](#_bookmark133)时序图所示

时钟



int\_味精\_有效

int\_ipfivf

int\_msix\_msg

int\_metadata

2'b00

2'b10

PF1-VF0

数据/ADDR 0

M0

2'b00

2'b10

PF1-VF0

数据/ADDR 1

M1

2'b00

2'b10

PF1-VF0

数据/ADDR 2

M2

2'b00

2'b10

PF1-VF0

数据/ADDR 3

M3

2'b00

msix\_enable（PF 1-VF 0）（来自int\_msix\_cap）

msix\_mask （来自int\_msix\_cap）

由于MSIX被禁用而

由于MSIX被屏蔽而中止

由于MSIX向量被屏蔽而中止

int\_msg\_done2'b00

2'b01

2'b00

2'b10

2'b00

2'b10

2'b00

2'b10

**图9.2：**MSI-X接口

* + 1. MSI接口

MSI接口用于客户端发送MSI中断，如[*图9.3*](#_bookmark135)时序图所示

时钟



int\_msg\_valid

2'b00

2'b01

2'b00

2'b01

2'b00

2'b01

2'b00

2'b01

2'b00

int\_ipfivf

int\_msi\_msg

int\_metadata

PF0

VEC3

M0

PF0

VEC3

M1

PF0 PF0

VEC 3 VEC 29

M2 M3

msi\_enable（PF0）（来自int\_msi\_cap）

msi\_mask（VEC3）（来自int\_msi\_cap）

已中止，因为MSI已禁用

由于载体被屏蔽而中止

已中止，因为矢量编号>已启用的矢量

int\_msg\_done2'b00

2'b01

2'b00

2'b10

2'b00

2'b10

2'b00

2'b10

**图9.3：**MSI界面

* 1. FLR处理
     1. HLS FLR处理

主机可以使用CFG-WR将函数或VF置于FLR。与该函数相关的所有事务都需要刷新。控制器将虚拟完成发送到来自正在进行FLR的功能的所有挂起的出站请求。主机可以向不同的功能发送多个FLR控制器还可以并行处理不同功能的多个FLR

以下是应用程序逻辑在接收到特定PF/VF的FLR时的预期如果在具有关联VF的PF上接收到FLR，则需要对所有关联 VF执行以下操作：

1. 中止该PF/VF的所有正在进行的事务
2. 停止发送任何出站事务。对于FLR下的功能，OB方向上的任何未决请求或完成都需要丢弃。
3. 等待出站方向上所有挂起请求*的良好完成*或*dummy-completion-with-flr-status*。这是为了在清除FLR状态之前，让此PF/VF的管道中的所有*np-request-transaction*向前移动
4. 为PF/VF断言client\_flr\_done

收到PF/VF的*client\_flr\_done*后，控制器在清除*flr\_in\_progress*后断言*controller\_flr\_done*

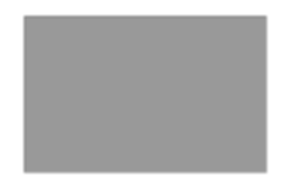
内部状态现在，应用逻辑可以恢复对该PF/VF的操作

任何收到的函数请求仍然处于静止状态，也就是说，客户端逻辑还没有以客户端FLR完成状态的形式进行确认

* + 所有的MEM/IO请求都会被悄悄丢弃。
  + 针对FLR下的函数的完成会被悄悄丢弃。
  + 所有针对FLR下函数的配置请求都会被悄悄丢弃。在这种情况下，有一个LM寄存器设置可用于响应CRS状态。

如果主机将FLR发送到具有活动VF的PF，则控制器将具有FLR状态的完成发送到具有待决NP请求的所有

 注：在这种情况下，将仅对PF执行FLR信号握手。



已收到的邮件

flr\_received\_ipfivf

flr\_client\_done\_ipdivf

flr\_ctrler\_done\_ipfivf

在客户端中止函数及其依赖VF上的所有事务后，由客户端逻辑断言。客户端还收到了其未决未完成请求的错误响应

在控制器断言此脉冲

**图9.4：**HLS FLR处理

# 第10章：电源管理

第10章：**电源管理**

* 1. 概述

本章介绍Cadence PCIe HPA控制器中实现的不同电源管理技术。有关如何使用可选功能的详细说明，对于规范强制功能及其在PCI系统中的使用，请参阅*PCI-SIG规范*。与电源管理特性相关的用户端信号和寄存器将在本节中进行说明。

* 1. 功能功率状态

当前版本的Cadence PCIe HPA控制器支持以下PCI功能电源状态：

* + - D0（未初始化和活动）。
    - D1 -有一个kwire来启用[SS 1]（可选）。
    - D3-热。

默认情况下，控制器将所有启用功能的**电源管理控制寄存器**的*No\_Soft\_Reset*位这意味着当处于D3-HOT功率状态时，功能的状态不会丢失，并且当功能被设置回D 0时，其寄存器不需要重新配置*PCI-SIG规范*建议所有功能均设置此位。

10.2.1功能电源状态变更握手界面

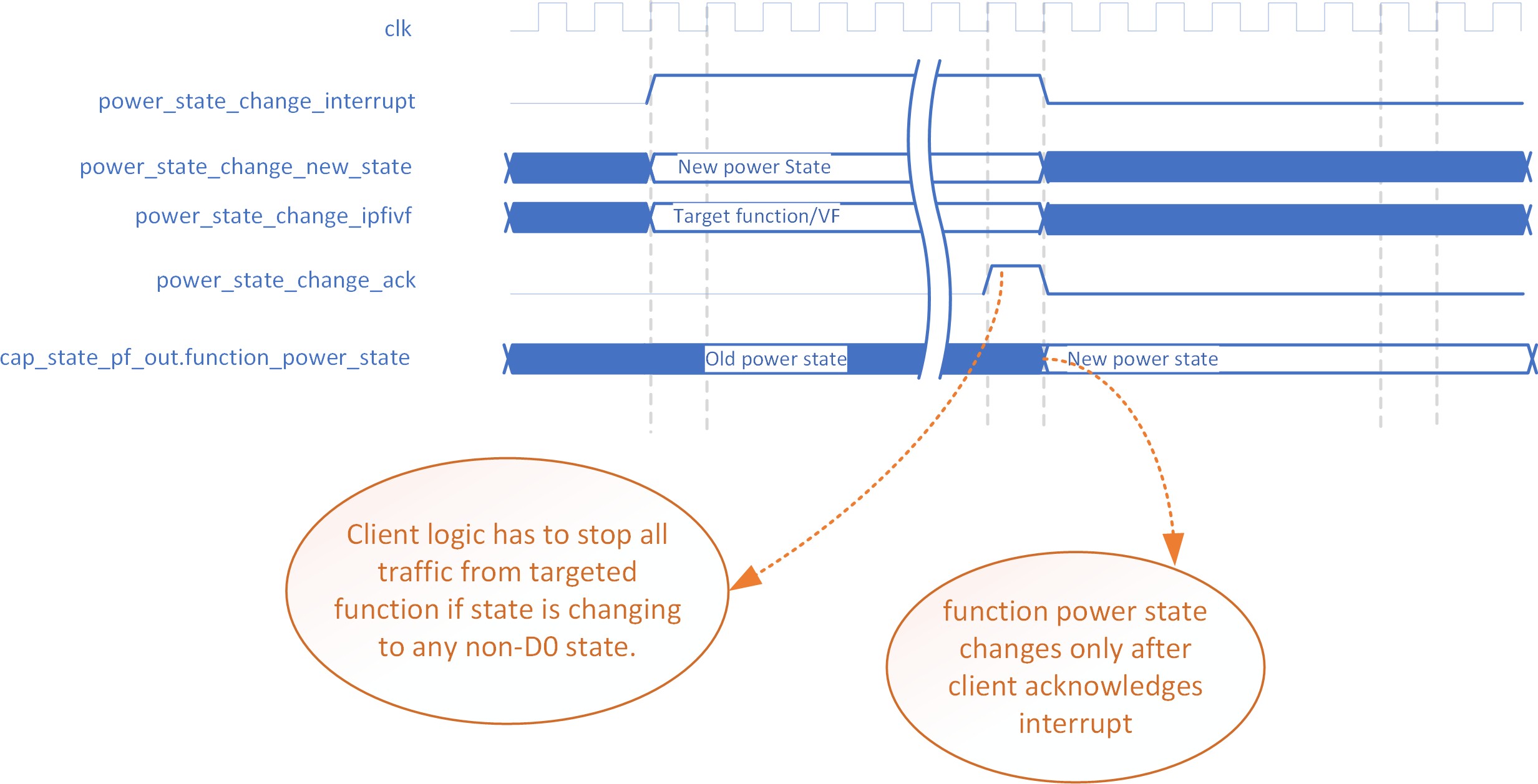
当电源管理控制和状态寄存器收到配置写入时，端点IP使用此接口更改其设备电源状态。时序图显示了成功更改电源状态所需的握手

从RP更改端点功能电源状态：

1. RP启动CFGWr以更改端点中的设备电源状态
2. 端点IP在接收到此CFgWr时断言*power\_State\_change\_interrupt*
3. 当*power\_State\_change\_interrupt*被断言时，控制器不响应任何其他CFG请求
4. 来自客户端逻辑*的ACK*更改该功能
5. 如果主机正在编程以移动到较低功率状态，则客户端逻辑必须仅在停止来自该功能的所有流量之后给出*ACK*

以下LM寄存器控制可用于控制哪个电源状态变化会产生中断：

* + **power\_state\_change\_interrupt\_cycle [0]**：当电源状态从D1或D3更改为D0时创建中断
  + **power\_state\_change\_interrupt\_cycle [1]**：当电源状态从D0或D3更改为D1时创建中断
  + **power\_state\_change\_interrupt\_cycle [2]**：当电源状态从D0或D1更改为D3时创建中断



**图10.1：**函数电源状态更改握手

* 1. PCIe链路电源状态
     1. L0链路电源状态

L0s的进入和退出是一个自主的过程。L0s状态在链路功率状态中具有非常低的退出延迟ASPM L0可以在*i\_PCIE\_cap\_struct*/*i\_link\_campus\_status*配置寄存器位**ASPMC**中启用或禁用。如果TX侧空闲，即在可编程时间段（*i\_regf\_ip\_pl*/*i\_pm\_cfg\_0*）内没有要传输的TLP和DLLP，则控制器自动启动进入ASPM L0。

* + - 1. 寄存器L0

L0 s进入超时可通过IP寄存器*i\_regf\_ip\_pl*/*i\_pm\_cfg\_0进行*编程。从L0到L0 s的转换发生在*i\_regf\_ip\_pl*/*i\_pm\_cfg\_0*字段**ASL 0 ST**中编程的时间段过去而没有传输TLP或DLLP之后。将*i\_regf\_ip\_pl*/*i\_pm\_cfg\_0*寄存器字段**ASL 0 ST**设置为0将禁用向L0 s状态的转换

* + 1. L1链路电源状态

L1链路功率状态以比L0s状态更多的延迟为代价提供更多的功率节省

* + - 1. 通过ASPM机制

ASPM L1是一种自治机制。进入和退出无需任何用户握手。ASPM L1可以在*i\_PCIE\_cap\_struct*/*i\_link\_campus\_status*配置寄存器位**ASPMC**中启用或禁用。如果TX端空闲，即在*i\_regf\_ip\_pl*/*i\_pm\_cfg\_1*寄存器字段**ASL 1 T**中的可编程时间段内没有TLP和DLLP要发送，则控制器自动启动进入ASPM L1。

以下序列显示了PCIe系统中ASPM L1条目的操作

1. 当链路TX空闲时（即，没有TLP和DLLP要传输），EP控制器开始在内部递增ASPM L1进入定时器如果客户端请求传输TLP，则定时器立即清零。
2. 当ASPM L1条目定时器达到*i\_regf\_ip\_pl*/*i\_pm\_cfg\_1*寄存器字段**ASL 1 T**的编程值时，EP控制器检查是否累积了足够的信用。
3. EP控制器阻止新的TLP，并通过发送*PM\_Active\_State\_Request\_L1*启动ASPM L1条目

DLLP到其传输通道上。

1. EP控制器持续传输*PM\_Active\_State\_Request\_L1 DLLP*，直至收到上游设备的响应
2. 上游组件必须立即响应请求，接受或拒绝请求。
3. 如果上游组件通过发送*PM\_Active\_State\_Nak*消息拒绝，则控制器中止ASPM L1条目并继续正常发送TLP。
4. 如果上游组件通过发送*PM\_Request\_Ack*DLLP来接受，则控制器（EP）将其TX置于电空闲并进入ASPM L1。
5. 上游组件还检测电空闲，并将其TX也置于电空闲。

注册集成详细信息

ASPM L1入口超时

ASPM L1进入超时可通过ip寄存器*i\_regf\_ip\_pl*/*i\_pm\_cfg\_1*字段**ASL 1 T**进行编程。字段“ASL1T”包含用于转换到ASPM L1功率状态的超时值（以16 ns为单位）。将其设置为0将禁用向ASPM L1电源状态的转换。

* + - 1. 通过PCI-PM软件兼容机制

以下顺序显示了将控制器配置为端点时PCI电源管理的操作

1. 假设控制器运行正常，所有功能处于D0激活状态。
2. 远程根端口写入所有功能的电源管理控制寄存器，以将功能转换为非D0电源状态。如果SRIOV已启用并受支持，根端口应从VF开始，然后对PF进行编程。
3. 要更改任何函数的电源状态，我们需要来自应用程序逻辑的确认，如[*功能电源状态更改握手接口一*](#_bookmark142)节所述。
4. 当控制器中所有启用的功能都处于非D0状态时，控制器通过发送*PM\_Enter\_L1*DLLP来启动链路功率状态到L1的转换
5. 在数据链路层握手之后，链路转换到L1状态。

注册集成详细信息

PM-L1重新进入超时

当链路处于L1状态并且所有控制器的功能处于D1或D3-HOT状态时，链路伙伴可以随时将链路从L1转换到L0。如果链路已经空闲了一个设定的时间间隔，并且控制器的功能仍然保持在D3-HOT状态，则控制器可以可选地发起重新进入L1。重新进入L1由ip寄存器空间中*i\_regf\_ip\_pl*/*i\_pm\_cfg\_2*中将此寄存器设置为非零值会导致

当经过等于该寄存器中编程的时钟周期数的延迟且无链路活动时，控制器启动返回L1的条目将此寄存器设置为0可防止重新进入L1。初始转换到L1，步骤[*当控制器中所有启用的功能都处于非D0状态时，控制器通过发送PM\_Enter\_L1 DLLP启动链路功率状态到L1的转换。*](#_bookmark146)不受此寄存器设置的影响。

* + - 1. L1退出触发事件

以下事件触发L1退出过程。

1. 电气怠速退出检测。
2. AXI或HLS接口上的新请求。
3. 当一个新的TLP被安排在内部传输时，例如，*ERR\_COR*消息。
4. 断言边带信号*low\_pwr\_client\_req\_exit\_l1*。
5. 使用根端口的*link\_control*
6. 为任何已发布/未发布/完成接口断言*hls\_ob\_activereq\**信号
   * + 1. ASPM输入

任何L1退出触发器事件都可以导致L1退出过程，并且不需要来自客户端的握手

* + - 1. PCI-PM软件兼容条目的L1退出过程

任何L1退出触发事件将使链路电源状态变为L0。然而，在到达L0之后，正常操作需要额外的根端口到端点握手

根端口启动的PM\_L1退出涉及的步骤包括：

1. 根端口启动CFG写入，将端点功能的电源状态更改为D0。
2. 此配置写入触发链路上的L1退出。
3. 端点DUT也从L1退出并以Cp1
4. 如果转换是从D3Hot取决于No\_Soft\_Reset控制器可以开始正常的数据传输或需要重新初始化。

端点启动PM-L1退出所涉及的步骤包括：

1. 当客户端在**PMSCR**中设置*PME\_STATUS*位时，端点向根端口发起*PM\_PME*消息以请求电源状态改变。
2. 此消息触发链路上的L1出口。
3. 收到此消息后，根端口启动CFG写入，将端点功能的电源状态更改为D0。
4. 端点被测设备响应Cpi。
5. 正常的数据传输可以在PCIe链路上进行。
   * 1. L1寄存器
        1. L1输入模式信息

ip寄存器空间中的寄存器*i\_regf\_ip\_pl*/*i\_pm\_dbg\_sts\_0*字段**L1 EM**给出关于最后L1进入模式的信息您将了解是否PM或ASPM模式被用于进入L1。这在每次L1进入操作之前复位

* + - 1. L1或L1-子状态退出触发事件信息

ip寄存器空间中的寄存器*i\_regf\_ip\_pl*/*i\_pm\_dbg\_sts\_0*字段**L1 EXCON**提供有关最后L1或L1子状态退出触发的信息此字段在每次L1入口操作之前重置。

* + - 1. 从客户端显式L1退出或阻止EP控制器的L1条目

客户端逻辑可以通过断言*low\_pwr\_client\_req\_exit\_l1来*触发显式L1（PCI-PM/ASPM）退出。该信号触发从L1或从L1子状态退出到L0HLS或AXI接口上的新请求也会在内部触发L1退出事件

此信号也可用于阻止端点控制器中的L1条目当客户端希望禁用L1条目以减少其未完成操作的延迟时，阻止L1条目非常有用

下图显示的时序图解释了此信号的两种用途。

恢复

堆芯



ltssm状态



L1.Idle

恢复

L0

low\_pwr\_client\_req\_exit\_l1

**图10.2：**使用low\_pwr\_client\_req\_exit\_l1退出L1

需要更少延迟

L1入口握手

堆芯



ltssm状态



L0

L0

L1.entry

L1.Idle

low\_pwr\_client\_req\_exit\_l1

图10.3：使用low\_pwr\_client\_req\_exit\_l1阻塞L1条目

* + 1. L1子状态

L1功率管理子状态ECN定义了一种可选机制，用于减少L1链路状态中的空闲功率这是通过在L1状态内定义新的子状态来实现的，以便于移除到phy的功率和到控制器的时钟。当链路进入L1时，由于PCI电源管理或ASPM，L1 PM子状态被启用。

当链路处于L1状态时，ECN提供了两种选择：L1.1子状态允许关闭时钟和大部分phy电源，但要求phy保持发送端的共模电压L1.2子状态无需维持共模电压，从而进一步降低空闲功耗。L1.1和L1.2状态都允许PHY中的电气空闲检测电路关闭。

L1 PM子状态使用*CLKCLK #*边带信号来控制时钟。*CLK\_CLK #*信号是由上游端口和下游端口共享的开漏低电平有效信号，并且可以通过将其驱动为低电平来断言这使能时钟发生器。当两端都取消断言其*CLKCLK #*输出时，内核时钟关闭

Cadence PCIe HPA控制器提供一个*低电平\_CLK\_CLK\_REQ\_IN\_N*输入和一个*低电平\_CLK\_REQ\_OUT\_N*输出，以实现三态*CLK\_CLK #*引脚。当低电平时，*low\_clair\_clkreq\_out\_n*输出使能三态驱动器，驱动*CLKCLK #*引脚，导致共享信号置位。*CLKCLK #*信号也可以由另一端的端口置位，将其驱动为低电平。此共享信号的状态由*Cadence PCIe控制器*通过*low\_clair\_clkreq\_in\_n*输入进行监控，如下图所示。



PCIe链路

L1\_PM\_SUBSTATE\_OUT

管接口

low\_blog\_clkreq\_in\_n low\_blog\_clkreq\_out\_n

low\_pwr\_phy\_ent\_l1\_x

low\_pwr\_phy\_ack\_l1\_x低功率物理负载空闲检测

低电平物理传输模式

公司简介

LTSSM

L1 PM子状态状态机

Cadence PCIe核心

PHY

CLK编号

对等组件

**图10.4：**L1 PM子状态实现框图

当内核时钟在L1.1和L1.2子状态下关闭时，单独的电源管理时钟（*pm\_CLK*）用于驱动L1 PM子状态状态机。无论链路的电源状态如何，该时钟必须始终处于活动状态。该时钟相对于控制器使用的其他时钟的相对相位没有要求

L1 PM Substates状态机还提供握手信号*low\_pwr\_phy\_ent\_l1\_x*和*low\_pwr\_phy\_ack\_l1\_x*，以使本地phy模块准备移除参考时钟。当状态机确定满足转换到L1.1或L1.2子状态的条件时，状态机在L1.0子状态下置位*low\_pwr\_phy\_ent\_l1\_x*输出。然后，在解除断言*low\_pwr\_clkreq\_out\_n*并进入L1.1或L1.2子状态之前，等待phy断言low\_pwr\_phy\_ack\_l1\_x。在L1.x退出期间，控制器将取消断言

*low\_pwr\_phy\_ent\_l1\_x*发送到PHY，并在转换到L1.0状态之前等待*low\_pwr\_phy\_ack\_l1\_x*这是为了确保PHY在进入L1.0之前完全工作并且时钟稳定对于L1.2的情况，当处于L1.2退出子状态时执行PHY握手。

 注意事项：

*low\_pwr\_phy\_ent\_l1\_x*的解除断言将PHY状态从L1子状态改变为L1.0，确保仅稳定的

在退出过程中，*将参考信号*

L1 PM Substates状态机提供输出信号*low\_logr\_phy\_logr\_idle\_det\_en*以通知本地PHY何时启用其接收侧上的电空闲检测电路。除了L1 PM子状态状态机处于L1.1、L1.2. Entry和L1.2. Idle子状态时，控制器在所有状态下均断言此输出。

L1 PM Substates状态机还提供输出信号*low\_pwr\_phy\_tx\_cmn\_mode\_en*，以启用PHY发送侧的当L1 PM Substates状态机处于L2.Idle子状态时，控制器取消断言该输出，并在所有其他时间断言该输出。

* + - 1. L1-子状态进入条件

L1-当链路处于L1中并且来自上游和下游组件的*CLKCLK #*被解除断言时，启动子状态进入过程控制器进入L1.1或L1.2，具体取决于L1 PM子状态控制寄存器中使能的内容。在进入L1.x子状态之前，控制器还在内部检查管道是否为空。

在两个L1子状态下，*low\_clkreq\_out\_n*均被解除断言当远程设备也解除断言CLKCLK #时，CORE时钟由用户域中的时钟控制器关闭。

* + - 1. L1-子状态退出条件

任一侧都可以发起从L1子状态的转换。远程端通过断言其*CLKCLK #*输出来启动L1 PM子状态退出，从而打开内核时钟。这也会置位控制器的*low\_clnr\_clkreq\_in\_n*输入客户端还可以启动L1的退出过程。

以下事件触发L1子状态退出过程，触发器分为两类

* + - * + 远程设备启动的退出触发器：

CLKREQ#断言

电气怠速退出检测（仅在进入L1.x子状态期间，

*low\_phr\_phy\_phr\_idle\_det\_en*被解除断言）

* + - * + 本地启动的退出触发器：

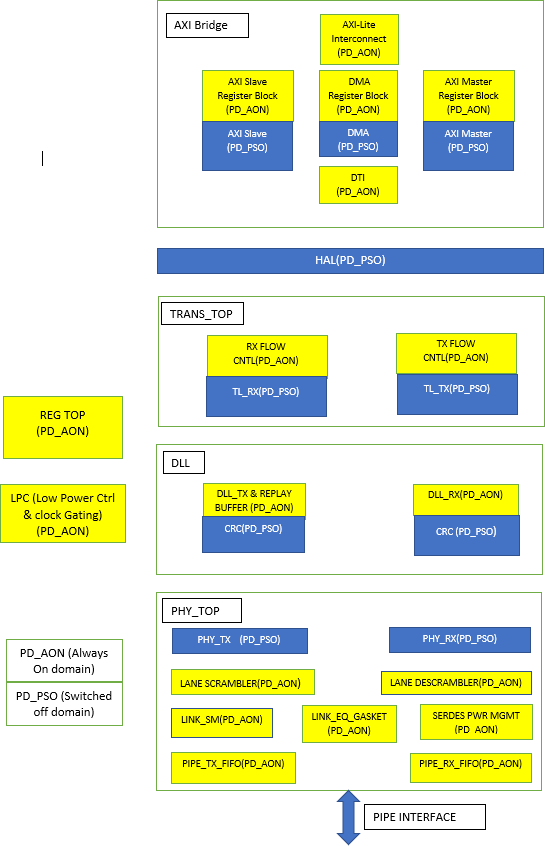
AXI或HLS接口上的新请求。

边带信号*low\_pwr\_client\_req\_exit\_l1*

为任何已发布/未发布/完成接口断言*hls\_ob\_activereq\**信号

L1-子状态退出触发器将链路从L1.x状态移动到L1.0，然后移动到L0。

下图显示了PCIe控制器中不同模块的电源域关联此图仅供参考



**图10.5：**PCIe控制器中不同模块的电源域关联

* + 1. L1.1操作

当L1.1进入条件为真且L1.2进入条件为假时，进入

* + - 1. L1.1进入和本地发起的退出

[*图10.6*](#_bookmark151)显示了L1.1进入和本地启动的退出过程。信号*low\_pwr\_client\_req\_exit\_l1*表示此图中的所有

公司简介



L1

L1

L1

L1

恢复

L1.0

L1.0

L1.1

L1.1

L1.1

L1.0

X

K

ltssm状态

low\_pwr\_l1\_pm\_substate\_out

low\_pwr\_phy\_ent\_l1\_

low\_pwr\_phy\_ack\_l1\_x

低电平时钟输出

low\_pwr\_clkreq\_in\_n low\_pwr\_client\_req\_exit\_l1

低功率物理负载空闲检测

低电平物理传输模式

芯线

**图10.6：**L1.1进入和本地启动退出

在发现满足进入L1.1子状态的条件时，L1 PM子状态机首先使用*low\_pwr\_phy\_ent\_l1\_x*和*low\_pwr\_phy\_ack\_l1\_x*信号执行与phy的握手，以使phy准备移除参考时钟。一旦phy断言了*low\_pwr\_phy\_ack\_l1\_x*，控制器就解除断言*low\_pwr\_clkreq\_out\_n*。如果链路伙伴也取消断言*其CLKREQ#*输出，则内核时钟变为非活动状态，控制器的*low\_clkreq\_in\_n*输入将取消断言。当检测到*low\_pwr\_clkreq\_in\_n*输入为高时，L1 PM Substates状态机转换到

任何本地L1子状态退出触发器都会将控制器带回L1.0状态。在退出期间，控制器置位*low\_clkreq\_out\_n*以打开内核时钟。这会导致*low\_clkreq\_in\_n*被置位。在感测到*low\_pwr\_clkreq\_in\_n*为低时，L1 PM子状态状态机通过解除断言*low\_pwr\_phy\_ent\_l1\_x*并等待phy以解除断言*low\_pwr\_phy\_ack\_l1\_x来*响应来执行与phy的另一次握手。该握手对于准备phy用于参考时钟的重新激活是必要的一旦该握手完成，phy转换回L1.0子状态。

公司简介



L1

L1

L1

L1

恢复

L1.0

L1.0

L1.1

L1.1

L1.1

L1.0

X

K

ltssm状态

low\_pwr\_l1\_pm\_substate\_out

low\_pwr\_phy\_ent\_l1\_

low\_pwr\_phy\_ack\_l1\_x

低电平时钟输出

低电平时钟输入n

low\_pwr\_client\_req\_exit\_l1 low\_pwr\_pwr\_idle\_det\_en

低电平物理传输模式

芯线

**图10.7：**L1.1进入和退出由链接伙伴发起

[*图10.7*](#_bookmark152)显示了当Controller所连接的链路伙伴发起从L1退出时的操作。当L1.2的进入条件不满足且L1.1的进入条件满足时，控制器从L1.0进入L1.1在完成与phy的*low\_pwr\_phy\_ent\_l1\_x/ phy\_ack\_l1\_x*握手之后，控制器解除断言*low\_pwr\_clkreq\_out\_n*。如果链路伙伴也解除断言其CLKREQ#输出，则核心时钟将变为不活动的，并且控制器的*低\_CLK\_CLK\_REQ\_in\_n*输入将被解除断言，从而导致L1 PM子状态状态机进入L1.1状态。

链路伙伴通过断言*其CLKCLK #*输出来启动链路从L1状态的转换。这将导致控制器的*low\_clkreq\_in\_n*输入置位在感测到*low\_pwr\_clkreq\_in\_n*为低时，L1 PM子状态机通过解除断言*low\_pwr\_phy\_ent\_l1\_x*并等待phy从L1.1退出来准备phy。

解除断言*low\_pwr\_phy\_ack\_l1\_x*。一旦该握手完成，L1 PM子状态状态机转换到L1.0子状态。同时，断言CLKCLK #会导致内核时钟变为活动状态。这使LTSSM能够从L1移动到Recovery。

* + 1. L1.2操作
       1. L1.2输入顺序

公司简介



ltssm状态

L1

L1

L1

ER关闭

TPOW

low\_pwr\_l1\_pm\_substate\_out

空闲

L1.2

n尝试

L1.2

n尝试

L1.2

L1.0

L1.0

low\_pwr\_phy\_ent\_l1\_x

low\_pwr\_phy\_ack\_l1\_x

低电平时钟输出

low\_pwr\_clkreq\_in\_n low\_pwr\_client\_req\_exit\_l1

低功率物理负载空闲检测

低电平物理传输模式

堆芯

**图10.8：**L1.2子状态进入操作

[*图10.8*](#_bookmark154)显示了当控制器配置为EP或RP时，L1 PM子状态机进入L1.2子状态的顺序如果满足L1.2的进入条件，则它首先使用*low\_pwr\_phy\_ent\_l1\_x*和*low\_pwr\_phy\_ack\_l1\_x*信号执行与phy的握手一旦phy断言了*low\_pwr\_phy\_ack\_l1\_x*，控制器就解除断言*low\_pwr\_clkreq\_out\_n*。如果链路伙伴也取消断言其CLKREQ#输出，则内核时钟将变为非活动状态，并且*low\_clock\_r\_clkreq\_in\_n*输入

控制器将被取消断言。L1 PM Substates状态机在其感测到L1.2.Entry时转换到L1.2.Entry。

*low\_pwr\_clkreq\_in\_n*输入高。当L1 PM子状态状态机处于L1.2.Entry子状态时，它会监视*low\_clkreq*\_in\_n输入，并在发现其被置位时转换回L1.0子状态如果*low\_clair\_*clkreq\_in\_n保持无效，则状态机将在L1.2.Entry子状态中停留TPOWER\_OFF间隔（设置为2微秒），然后转换到L1.2.Idle子状态。

* + - 1. L1.2退出顺序

当L1 PM子状态状态机处于L1.2.Idle子状态时，本地客户端或链路伙伴可以发起链路从L1子状态的转换。下图说明了本地客户端存在退出触发信号*low\_pwr\_client\_req\_exit\_l1*表示[*图10.9*](#_bookmark155)中的所有本地退出触发器

公司简介



L1

L1

L1恢复

MinimumTL1.2

T通电

L1.2进入L1.2空闲

L1.2怠速

L1.2出口

L1.2出口

L1.0

ltssm状态

low\_pwr\_l1\_pm\_substate\_out

low\_pwr\_phy\_ent\_l1\_x

low\_pwr\_phy\_ack\_l1\_x

低电平时钟输出

low\_pwr\_clkreq\_in\_n low\_pwr\_client\_req\_exit\_l1

低功率物理负载空闲检测

低电平物理传输模式

堆芯

**图10.9：**本地启动的退出

任何一个本地退出触发器都会启动L1.2退出过程。控制器首先置位*low\_clkreq\_out\_n*以打开内核时钟。这会导致*low\_clkreq\_in\_n*被置位。在感测到*low\_clr\_clkreq\_in\_n*为低时，L1 PM子状态状态机转换到L1.2.Exit子状态。

当处于L1.2.Exit子状态时，L1 PM子状态状态机执行与phy的*low\_pwr\_phy\_ent\_l1\_x/ phy\_ack\_l1\_x*握手，以使phy为时钟的重新引入做好准备，并且随后转换回到L1.0子状态。L1 PM子状态状态机必须在L1.2.Exit子状态中保持最小间隔TPOWER\_ON。该间隔的持续时间由“i\_regf\_L1\_PM\_cap\_struct/i\_L1\_PM\_ctrl\_2”寄存器中TPOWER\_ON值和比例参数的设置决定根据这些寄存器字段的设置，间隔可以在0到3100微秒之间变化。

公司简介



L1

L1

L1恢复

MinimumTL1.2

T通电

L1.2入口L1.2空闲

L1.2怠速

L1.2出口

L1.2出口

L1.0

ltssm状态

low\_pwr\_l1\_pm\_substate\_out

low\_pwr\_phy\_ent\_l1\_x

low\_pwr\_ack\_l1\_x

低电平时钟输出

low\_pwr\_clkreq\_in\_n low\_pwr\_client\_req\_exit\_l1

低功率物理负载空闲检测

低电平物理传输模式

堆芯

**图10.10：**link\_partner发起的退出

[*图10.10*](#_bookmark156)显示了当Controller所连接的链路伙伴发起从L1退出时的操作当处于L1.2.Idle子状态时，链路伙伴通过断言

它*的CLK #*输出。这将导致控制器的*low\_clkreq\_in\_n*输入置位。在感测到*low\_clr\_clkreq\_in\_n*为低时，L1 PM子状态状态机转换到L1.2.Exit子状态（在满足L1.2.Idle中停留最少4 us的要求然后，在完成与phy的握手以重新启用其时钟之后，L1 PM子状态状态机转换到L1.0子状态，并在*TPOWER\_ON*的最小间隔内停留在L1.2.Exit子状态。

* + 1. 注册L1-Substate
       1. 规范定义的L1子状态寄存器

[*表10.1*](#_bookmark158)提供了程序规范定义寄存器的参考。有关寄存器的完整列表，请参阅用户指南中的*寄存器*从PCIe链路访问时，带有前缀port\_\* 的寄存器和能力寄存器是只读的这些寄存器可通过ip寄存器访问接口写入。这些寄存器需要初始化以匹配PHY和系统电气特性。标准系统初始化软件使用这些寄存器对根端口和端点的L1子状态能力空间中的RW控制寄存器进行编程。

表10.1：L1子状态寄存器

|  |  |  |  |
| --- | --- | --- | --- |
| 寄存器名称 | 价值选择指南 | 谁更新 | 错误价值观 |
| L1PrtPvrOnVal和L1PrtPvrOnScale | 端口的PHY T\_POWER\_ON值。例如，这是Cadence PHY | 本地客户端的控制器和物理层初始化固件。 | PCIe系统初始化软件使用该值在**T\_power\_on**寄存器中编程。 |
| L1PwrOnVal和L1PwrOnSc | 最大值（EP Port\_T\_power\_on，RC Port\_T\_power\_on） | 主机上的PCIe系统初始化 | 低于所需值的值使一个设备驱动到一个无动力远程设备。这可能导致link\_down事件，并且LTSSM移动到检测状态。 |
| L1PrtCmMdReStrTime | 端口的PHY建立所  共模有效期间  TS1的传播。例如，在Cadence的PHY中，该值是t\_common\_mode。 | 本地客户端的控制器和PHY初始化固件。 | IBM软件使用此值进行编程  共模恢复时间寄存器。 |
| L1CmMdReStrTime | 最大值（RC端口共模恢复时间，EP  端口共模  restore\_time） | PCIe系统初始化软件 | 低于所需值的值使一个设备驱动到一个无动力远程设备。这可能导致link\_down事件，并且LTSSM移动到检测状态。 |

表10.1：L1子状态寄存器

|  |  |  |  |
| --- | --- | --- | --- |
| 寄存器名称 | 价值选择指南 | 谁更新 | 错误价值观 |
| L1ThrshldVal和L1ThrshldSc \* | 仅当启用LTR且启用ASPM L1.2时才使用此选项。这是在存在L1.2的情况下完成请求所面临的最差情况延迟。请参考PCIe规范的LTR部分，以了解LTR机制的用法。  LTR\_L1\_2\_threshold = Service\_request\_latency + 2 \*  （TL1.2 + TL1O\_REFCLK\_ON  + TP1\_to\_P0 + TCOMMON- MODE +2us）;   * Service\_request\_latency：：根端口端客户端响应读取请求或接受WR请求的 * TL1.2 -根据规范，停留在L1.2的最短时间。这是4us * TL1O\_REFCLK\_ON -退出L1.2时，对Refresh的CLKSTART #断言有效。这是TPOWER\_ON +系统使能参考时钟所需的额外时间;   + TP1\_to\_P0 -PHY改变功率状态所   从P1到P0 TCOMMONMODE   * + 与common\_mode\_ restore\_ time寄存器相同;   + 2us -考虑控制器和握手中的延迟的裕度;   + 表达式中需要与2相乘，以说明请求和/或完成可能需要的L1.2退出。 | PCIe系统初始化软件 | ASPM L1.2条目仅在端点的LTR要求大于此阈值时发生。阈值编程不正确可能导致端点请求出现意外延迟。 |

 注意事项：

* + - * + 如*PCIe规范*中所定义的，ASPM L1.1条目不会自动检查延迟阈值要求。因此，端口PHY（TL1.1\_TO\_REF\_CLK\_ON + TP1\_to\_P0）必须小于客户端应用的可能最小延迟要求。TL1.1\_TO\_REF\_CLK\_ON是在L1.1中断言*CLKCLK #*后打开参考时钟的时间。例如，这是tP1.1\_to\_P1+系统为Cadence PHY打开参考时钟所TP1\_to\_P0是PHY将功率状态从P1改变到P0所花费的时间如果这个延迟更大，则需要关闭L1.1以满足客户端应用的LTR要求
        + 如果您选择了内部电源关闭功能，则必须将12个PM\_CLK周期添加到LTR\_L1\_2\_thresholds。
      1. L1子状态进入延迟

从基本L1进入L1子状态可以通过ip寄存器*i\_regf\_ip\_pl*/*i\_pm\_cfg\_5*中的字段**L1 SED**进行延迟。控制器在等待此延迟到期时响应L1退出事件。

* + - 1. 从客户端显式退出L1子状态或阻止L1子状态进入

客户端逻辑可以通过断言*low\_pwr\_client\_req\_exit\_l1*来触发显式L1子状态退出。如果控制器已经处于L1子状态，则该信号触发从L1子状态到L0的确保在L1子状态关闭时使用此信号

时钟驱动新的请求到HLS或AXI接口。强烈建议客户端实现硬件逻辑，以便在从客户端发起新的硬件逻辑减少了固件或软件驱动程序的开销。如果您希望阻止l1子状态条目以减少任何挂起的未完成操作的延迟，则此信号也很有用。[*图10.11*](#_bookmark159)显示了该信号的使用。

 注：图中的L1.x可以是L1.1或L1.2。

堆芯



L1.x退出过程

L1出口过程

L0

非活动

L1.0

L1.x.exit

L1.x

L1.Idle

L1.Idle

ltssm状态

low\_pwr\_l1\_pm\_substate\_out

low\_power\_client\_req\_exit\_l1

**图10.11：**使用low\_pwr\_client\_req\_exit\_l1退出L1.x

* + 1. L1子状态
       1. PM\_CLK频率选择

在L1-子状态下，可以关闭*pd\_pso\_core\_socket*。L1子状态机需要一个单独的时钟*PM\_CLK*才能运行。在配置器中选择此时钟的默认频率客户端固件需要编程

ip寄存器中的*i\_regf\_ip\_pl*/*i\_pm\_cfg\_7*字段**PMCFS**，用于更改*PM\_CLK*频率。仅当LTSSM不在L1中时，才支持*PM\_CLK*频率的更改

仅当在PCIe配置空间中启用L1子状态功能时，才需要Pm\_REQ

* + - 1. 内部电源关闭

低功率链路状态在*PCI-SIG规范*中定义。这是链路的低功率状态，其中时钟和PCIe链路相关电路关闭。PCIe控制器具有内部电源关断功能，可进一步提高节能效果。内部电源关断可减少漏电流，并提高L1.2链路状态下的省电性能。这是在基于CPF的潮流的帮助下完成的。进入和退出内部电源关闭不需要来自客户端的任何特殊握手有调谐寄存器和边带信号来改变PCIe控制器的默认行为

进入L1.2并关闭

在进入基本L1状态后，如果满足以下条件，控制器进入L1.2子状态并断电

1. 进入L1.2子状态的条件为真。有关这些条件的详细说明，请参阅*PCI-SIG规范*
2. 到客户端侧路径缓冲区的PCIe链路为空。
3. 没有来自PCIe链路和客户端的未决数据包

控制器在基本L1状态下等待以满足这些条件。一旦满足这些条件，控制器进入L1.2子状态，并在L1.2空闲状态下关闭电源。在等待期间，控制器通常响应L1退出条件。如果在IP寄存器中禁用电源关断，则控制器从L1.0进入L1.2，而不等待这些电源关断条件。

退出电源关闭模式

退出L1.2子状态的条件也会触发从电源关闭模式退出更多信息，请参见[*L1子状态*](#_bookmark149)部分

PCIE控制器

PCIe控制器电源管理逻辑分为两个电源域：

* + 电源关闭电源域（PD\_PSO）
    - 包括控制器的子模块，可在L1子状态期间断电
    - 包含电源管理逻辑、LTSSM和PIPE接口FIFO。
    - 即使在L1子状态期间，这些开关也始终处于ON状态。
    - 电源管理逻辑在名为“PM\_CLK”的独立时钟上运行。
    - 当处于L1子状态时，LTSSM和PIPE接口FIFO模块的时钟将关闭，但电源将打开以保持这些模块的状态。
  + 始终开启电源域（PD\_AON）
    - 包含电源管理逻辑、LTSSM和PIPE接口FIFO。
    - 即使在L1子状态期间，这些开关也始终处于ON状态。
    - 电源管理逻辑在名为“PM\_CLK”的独立时钟上运行。
    - 当处于L1子状态时，LTSSM和PIPE接口FIFO模块的时钟将关闭，但电源将打开以保持这些模块的状态。

 注意：准确的域信息在CPF文件中可用。

提供电源关闭方案

基于CPF的功率流和内部功率及隔离控制器

关闭和打开电源和隔离的顺序由内部控制逻辑生成在此模式下，用户不需要握手只需遵循CPF功率流即可实现功率关断。

基于CPF的功率流，具有旁路模式下的内部电源和隔离控制器

边带信号LP\_CTRL\_BYPASS\_ENABLE使能电源和隔离控制器的旁路模式在此模式下，关闭和打开电源和隔离的顺序由用户逻辑生成。用户可使用LP\_CTRL\_\* 信号将电源序列事件传递至控制器。用户仍可使用输送CPF功率流。在旁路模式下，控制CPF流量的信号被复用以使用来自用户的LP\_CTRL\_\* 信号。

[*图10.12*](#_bookmark161)说明了电源和隔离控制器旁路模式。



内部功率

侧控制器

LP\_CTRL\_\* 输入信号

LP\_CTRL\_BYPASS\_ENABLE

内部信号

CPF Flow使用

确认

通电和断电

CTRL\_POWER\_SUSPEND\_STOPCTRL\_POWER\_RECOVER\_STOP

**图10.12：**内部电源和隔离控制器的旁路模式

LP\_LP\_

L1.x状态机

用户电源和隔离控制器

控制器RTL顶部-仅电源管理逻辑

时钟门控单元和复位同步器

控制器IP中存在时钟选通锁存器，应在后rtl工具流程中考虑

这些锁存器用于在关闭电源域之前关闭内部时钟控制器IP中存在的时钟门控锁存器实例被命名为：**.i\_lp\_code.u\_code\_gating\_cell\_**

控制器IP中存在重置同步器，这应该在post-rtl工具流中考虑这些用于同步基本重置和通电序列期间内部生成的重置。

控制器IP中存在的重置同步器实例命名为：**.i\_lp\_同步器.u\_reset\_sync\_**

关闭外部SRAM

外部SRAM也可以通过使用PD\_PSO电源域关断条件来关断。请参阅*CPF文件*以获取关闭PD\_PSO的条件另一种方法是使用*low\_bandwidth\_power\_suspend\_req*，

*low\_bandwidth\_power\_recover\_req*边带信号，以触发SRAM断电和上电序列。这些信号仅在满足内部电源关断条件时才在L1.2中置位该信号上的零到一个转变指示相应序列的触发。在正常模式下，用户的SRAM上电和断电序列预计将在4*个PM\_CLK*周期内完成。

控制电源关断

以下调谐寄存器控制控制器的默认行为

关闭电源关闭功能

电源关闭功能可以通过使用本地管理空间中的字段“低功耗调试和控制寄存器1/启用L1.2中的电源关闭”来关闭如果电源关闭被关闭，控制器进入L1.2子状态，无需等待电源关闭条件。如果该字段被设置，则在L1.2子状态期间不关闭内部电源。

等待断电条件时的断电机制

超时机制可以通过本地管理空间中的字段“Low Power debug and control Register 1/Low Power while waiting for RX IDLEand OUTSTANDING Idle”超时将迫使控制器进入L1.2子状态，而无需电源关闭机制。L1.2退出后，控制器正常响应来自PCIe链路和客户端的挂起数据包。客户可以使用此功能，如果系统中存在一些需要很长时间的响应条件

超时将使控制器进入L1.2并在等待响应时节省功率。本地管理空间中的状态字段“在RX和未完成等待期间发生低功率调试和控制寄存器1/寄存器1”给出关于在该超时之后是否触发L1.2条目的指示。

控制电源切断

内部隔离和电源切断控制器的旁路模式

下表列出了旁路模式所需的信号

|  |  |  |  |
| --- | --- | --- | --- |
| 接点名称 | 方向 | 时钟 | 描述 |
| 低电平旁路使能 | 在 | 静态 | 将隔离和电源关断控制器置于旁路模式。在旁路模式下，客户端逻辑完全控制隔离和电源关断顺序。如果不需要旁路模式，则将此信号连接到“0”。 |
| 低功耗电源暂停请求 | 出来 | PM\_CLK | 这是用户启动power\_suspend-sequence的指示。用户必须声明  如果LP\_CTRL\_BYPASS\_ENABLE为真，则在完成断电序列后LP\_CTRL\_POWER\_SUSPEND\_ACK。此信号可用于SRAM关闭或其他用户侧逻辑关断用户.“0”到“1”的转换是power\_suspend序列请求。如果LP\_CTRL\_BYPASS\_ENABLE为假，预计用户将在4 PM\_CLK内完成断电序列。 |
| 低功耗暂停确认 | 在 | PM\_CLK | 在旁路模式下完成断电序列后，用户必须断言此信号如果LP\_CTRL\_BYPASS\_ENABLE为真，则仅由控制器使用。 |

|  |  |  |  |
| --- | --- | --- | --- |
| 接点名称 | 方向 | 时钟 | 描述 |
| 低功耗恢复请求 | 出来 | PM\_CLK | 这指示用户启动power\_recover- sequence。  如果LP\_CTRL\_BYPASS\_ENABLE为真，则需要在完成上电序列后断言LP\_CTRL\_POWER\_RECOVER\_ACK该信号可用于SRAM上电或其他用户侧逻辑上电。“0”到“1”的转换是power\_recover序列请求。如果LP\_CTRL\_BYPASS\_ENABLE为假，则预期在4 PM\_CLK内完成上电序列。 |
| 低功耗恢复确认 | 在 | PM\_CLK | 在旁路模式下完成上电序列后，需要断言此信号如果LP\_CTRL\_BYPASS\_ENABLE为真，则仅由控制器使用。 |
| low\_blog\_clog\_state\_ret\_save | 在 | PM\_CLK | 此信号中的脉冲在电源关闭前保存内部状态。仅当LP\_CTRL\_BYPASS\_ENABLE为真时，控制器才使用此选项。  注意：此配置中不使用*此参数用户可以将此参数绑定到“0”。* |
| low\_blog\_clog\_state\_ret\_restore | 在 | PM\_CLK | 该信号中的脉冲恢复内部状态。仅当LP\_CTRL\_BYPASS\_ENABLE为真时，控制器才使用此选项  注意：此配置中不使用*此参数用户可以将此参数绑定到“0”。* |
| 低功耗 | 在 | PM\_CLK | 电源使能阶段1：该信号控制电源岛上电的阶段1。两阶段启用  该状态机支持该过程，因此可以错开电源岛上的组件的上电，以防止过度的电流消耗。  注：*这是一个****低电平有效****信号。*   * 1'b0：电源启用 * 1'b1：电源禁用   仅当LP\_CTRL\_BYPASS\_ENABLE为真时，控制器才使用此选项。 |
| 低功耗隔离 | 在 | PM\_CLK | 电源隔离使能：激活此信号可启用CPF流程中所述的电源岛隔离功能  仅当LP\_CTRL\_BYPASS\_ENABLE为真时，控制器才使用此选项。 |
| 低电平复位 | 在 | PM\_CLK | 电源复位：该信号用于在被控制的电源岛被禁用后打开时，对任何需要复位的逻辑进行复位。  仅当LP\_CTRL\_BYPASS\_ENABLE为真时，控制器才使用此选项。 |

保留和断电顺序>

[*图10.13所*](#_bookmark162)示为电源控制器旁路模式下low\_cnr\_cnr\_\* 信号的预期时序关系。通过置位LP\_CTRL\_BYPASS\_ENABLE使能电源控制器旁路。Power\_suspend\_ack完成power\_down序列，如下图所示应用程序逻辑可以在每个步骤之间插入延迟

LP\_CTRL\_prefix在时序图信号名称中省略L1.2.仅在断电序列完成后才处理空闲退出触发

**PM\_CLK**

L1\_PM\_子状态

L1.0

**T\_power\_off**

L1.2.进入L1.2.空闲

电源暂停请求

电源隔离使能

状态\_ret\_save

电源\_en\_n

电源暂停确认

电源复位n

通电和恢复顺序

**图10.13：**断电顺序

[*图10.14所*](#_bookmark163)示为电源控制器旁路模式下*low\_cnr\_cnr\_\**信号的预期时序关系。电源控制器旁路通过置位*low\_bypass\_enable*使能。时序图显示了上电顺序。在时序图信号名称中省略了*low\_blog\_clog前缀*

**PM\_CLK**

L1\_子状态

L1.2.Idle

**L1.0**

出口

L1.2.

T\_pow

厄昂

电源恢复请求

电源复位n

电源\_en\_n

状态\_ret\_restore

电源隔离

电源恢复确认

**图10.14：**电源恢复顺序

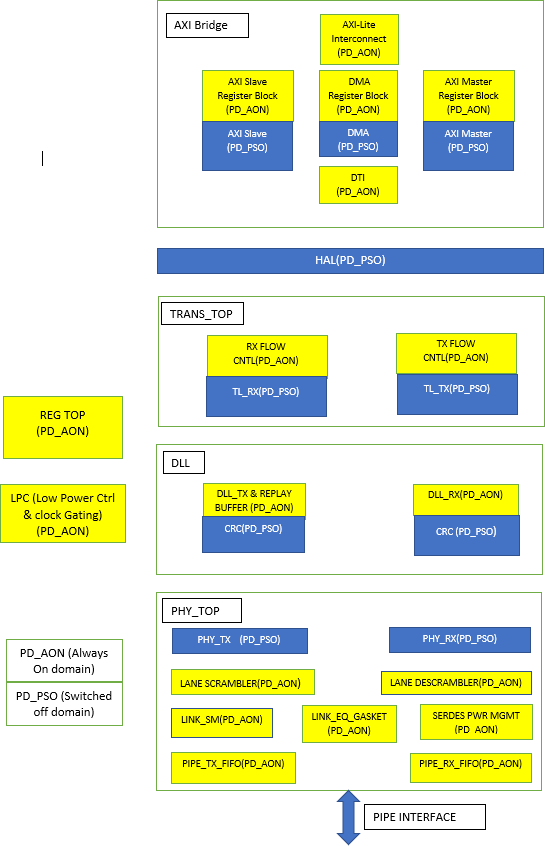
电源域

PCIe控制器电源管理逻辑分为两个电源域。

* + 电源关闭电源域（PD\_PSO）
* 包括控制器的子模块，可在L1子状态期间断电
  + 始终开启电源域（PD\_AON）
* 包含电源管理逻辑、LTSSM和PIPE接口FIFO。即使在L1子状态期间，这些开关也始终处于ON状态

电源管理逻辑在称为“PM\_CLK”的独立时钟上运行

下图显示了PCIe控制器中不同模块的电源域关联下图是指示性的，准确的域信息可在CPF文件。



**图10.15：**电源域关联

* + 1. L2和L3
       1. L2和L3入门程序

L2和L3都是不向整个控制器供电的省电状态这种状态下的电源管理通过RP发送PME\_Turn\_Off消息开始进入L2/L3的握手。

端点所需的步骤

1. 当EP收到来自RP的PME\_Turn\_OFF消息时，控制器设置以下信号。应用程序逻辑（AL）可以选择其中一个选项来获得此事件的通知PCIe规范允许RP在D0和非D0状态下发送PME\_Turn\_off消息。
   1. low\_pwr\_l2\_pme\_turn\_off\_received
   2. local\_interrupt和interrupt\_signals\_sideband[32]，主要反映“本地错误状态寄存器”的位[0

1”。客户端可以使用“本地错误掩码寄存器1”对此进行掩码，在这种情况下，此处列出的任何信号都不会被设置。

1. 根据PCIe规范要求，AL需要在10 ms内确认RP的断电请求AL可能具有正在进行的未完成的分组，其可能需要在确认PME关闭请求之前刷新。一旦准备就绪，AL可以通过以下两种方式中的任何一种来设置此确认：
2. 将low\_pwr\_l2\_pme\_turn\_off\_ack设置为一个周期。
3. 如果使用Local Error方法，则向“Local Error Status Register 1”的位[0]写入1以将其清除。这将在内部创建类似于上面的“a”的ACK脉冲
4. 一旦AL设置了ACK，控制器将等待在“i\_pm\_cfg\_7[31：24]：PM L2条目OB分组刷新命令”中等待待刷新的OB分组，然后它调度PME关闭ACK消息并发布开始L2条目的DLLP握手。
5. 为了确定成功进入L2，AL可以查看“cap\_state\_common[53：50]"是否等于4”h8，“cap\_state\_common\_change\_pulse”可以用作触发器以检查是否相同。
6. 然后，AL可以关闭时钟和电源。

RP所需步骤

1. 在发送PME关闭消息之后，控制器等待来自端点的PM进入L2 DLLP
2. 一旦接收到这些DLLP，控制器在出站管线为空或编程超时后启动进入L2/L3的步骤

出于调试目的，有ip-mgmt寄存器可用于在接收DLLP时延迟或阻止进入L2请参考“i\_pm\_cfg\_4/Block PM\_Request\_ACK for Enter L23 DLLP”和“i\_pm\_cfg\_7/HLS OB Deactivate Timeout”。

* + - 1. 从L2、L3状态

PCIe控制器支持使用唤醒机制的系统。

PCIe控制器的PME上下文应在向根复合体发送PME\_TurnOff确认消息之前由应用逻辑捕获。在确认pme\_turnoff之前，客户端逻辑必须存储来自本地管理的控制器的总线号和设备号。客户端逻辑应指定重新向控制器供电且链路达到L0时发送以下序列描述了支持*WAKE#*的过程。

1. 假设控制器位于L2。客户端逻辑使用由Vaux或全功率供电的客户端应用程序逻辑来维护PME上下文和路由器ID
2. 现在，客户端逻辑可以决定唤醒控制器。客户端逻辑应驱动*WAKE#*带外信号，以向电源管理控制器发出信号，通知控制器需要重新供电。

 注意：*WAKE#*信号位于PCIe控制器外部，不被PCIe控制器使用。

1. 控制器重新通电
2. 恢复供电后需要硬复位，以使链路恢复到L0

a客户端逻辑然后经由ip寄存器访问接口将任何PME上下文恢复到控制器寄存器b通过对位20进行编程，防止控制器自动发送PM\_PME消息

*disable\_pme\_message\_on\_pm\_status*为“1”（在IP寄存器PME服务延迟寄存器中）。

c在设置*PME状态*之前，*disable\_pme\_message\_on\_pm\_status*位需要编程为“1”

位为“1”。

1. 然后，客户端逻辑使用在进入L2/L3之前捕获的转发器ID通过客户端接口发送*PM\_PME*消息
2. 根联合体然后可以执行对控制器的配置写入，以便将设备从D3热状态移动

# 第11章综合与物理设计指南

第11章**综合与物理设计指南**

* 1. 合成设置

集成电路综合策略是将设计中的每个IP与所有其他IP隔离，单独进行综合和优化。然后依赖于IP级合成约束的以下特性，以确保在顶层不需要因此，顶层网表集成任务被简化为所构成的网表的IP简化

表11.1：可回收物

|  |  |
| --- | --- |
| 类别 | 描述 |
| 合成 | 已使用Genus Version*/genus/162/16.21.000*运行合成 |
| 图书馆 | 指向您的特定库。 |
| 约束 | 在*dbPcie\_ios\_top/*目录下有一个*synth/constraints*目录，其中包含约束文件：cdnhpa\_ips\_default.func.sdc。 |
| RTL/库 | *tcl*和*setup.tcl*指向我们的示例合成运行。确保您更改了指针，以便为我们的环境选择正确的rtl和库。 |
| 脚本 | 以下脚本用于合成：synth/scripts/genus/genus\_synth.tcl |
| 命令 | 运行合成使用，  ./运行\_phys.csh-synth |

 注意事项：

* + - 如果您使用Genus作为合成工具，请确保优化开关处于打开状态。此外，确保没有模块保留属性
    - 如果您使用Synopsys DC作为合成工具，请确保“删除冗余逻辑”开关处于打开状态。
  1. 技术特定模型

您可以将下面的组件提供的*/hdl/hdl\_src/reuse\_components*目录替换为保留相同文件名的特定于技术的组件。

* + - cdnsdru门控锁存器合成器
    - cdnsdru\_scan\_mux\_synth
    - cdnsdru\_reset\_sync\_synth
    - cdnsdru\_databook c\_synth\_sr
    - cdnsdru\_databook\_synth\_ar
    - cdnsdru\_mux\_synth
  1. CPCS模块

CPCS的时钟复用器*cpcs\_synth\_mux\_mux.v*是一个重用模块，位于*rtl/cpcs/*文件夹中。确保将*cpcs\_synth\_mux\_mux.v*替换为特定于技术的库模块。cpcs*\_synth\_mux\_mux.v*模块的位置

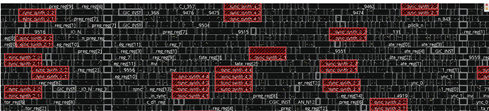
<trunk>/rtl/cpcs/cpcs\_synth\_mux\_mux.v

* 1. 在物理设计

异步FIFO使用跨时钟域的格雷码向量来传递读指针和写指针。与时钟相比，该向量的位之间的偏斜必须最小PNR（放置路由）工具需要小心处理此向量同步器块，以免在向量位之间添加偏斜。

格雷码矢量同步器需要特殊处理。

* + 每个同步器组都被视为需要特殊处理的结构化数据路径
  + 推荐使用具有最低Vt的亚稳定性硬化触发器一些铸造厂自己提供特殊的同步器单元
  + 如果使用单个触发器创建同步器组，建议使用对齐放置。触发器也将按列分组，这大大简化了时钟布线，因为如果触发器是相同类型，则时钟引脚将对齐
  + 使用相同的时钟缓冲区或相同的时钟门控单元驱动组中的触发器，并将它们靠近放置以优化路由
  + 目标触发器的Q和D之间没有组合逻辑PD（物理设计）不能在同步器组中的触发器之间插入缓冲器



**图11.1：**格雷码指针同步器的示例引导放置

PCIe控制器IP为异步FIFO使用公共重用块时钟域同步器在整个设计中也很客户应将同步器更换为亚稳态强化库特定单元。

可以通过搜索以下实例模式来获得异步FIFO灰度编码向量

* + \*ptr\_gray**[\*]**.sync\_cell.cdns\_syncflop\_inst\*
  + \* 同步 **\*** .cdns\_syncflop\_inst.\*.
  + \*ptr**.\*** .cdns\_syncflop\_inst.\*

IP中使用了以下ADC FIFO模块列出了每个模块的同步器导线输入的源导线此处列出的内容仅供参考。

* + <prefix>cpcs\_rctc
  + wr\_ptr\_gray-> i\_data\_wr\_ptr\_gray[\*].sync\_cell.cdns\_syncflop\_inst.\*.u\_cdnsdru\_databasec\_synth.d\_in
  + <prefix>cpcs\_128b\_rctc\_32b
  + wr\_ptr\_gray-> i\_phy\_data\_sync\_wr\_ptr\_gray[\*].sync\_cell.cdns\_syncflop\_inst.\*.u\_cdnsdru\_databasec\_synth.d\_in
  + <prefix>pipe\_fifo
  + write\_addr\_gray-> write\_addr\_sync.\*。cdns\_syncflop\_inst.\*。u\_cdnsdru\_databasec\_synth.d\_in
  + read\_addr\_gray-> read\_addr\_sync.\*.cdns\_syncflop\_inst.\*。u\_cdnsdru\_databasec\_synth.d\_in
  + <prefix>pipe\_tx\_fifo
  + write\_addr\_gray-> write\_addr\_sync.\*。cdns\_syncflop\_inst.\*。u\_cdnsdru\_databasec\_synth.d\_in
  + read\_addr\_gray -> read\_addr\_sync.\*. cdns\_syncflop\_inst.\*。u\_cdnsdru\_databasec\_synth.d\_in
  + <prefix>afifo
  + afifo\_wclklog.wrPtr-> afifo\_rclklog.sync\_wr\_ptr.\*。cdns\_syncflop\_inst.\*。u\_cdnsdru\_databasec\_synth.d\_in
  + afifo\_rclklog.rdPtr-> afifo\_wclklog.sync\_rd\_ptr.\*。cdns\_syncflop\_inst.\*。u\_cdnsdru\_databasec\_synth.d\_in
  + <prefix> prefix\_fifo
  + wptr\_gray-> i\_cdnsdru\_databook c\_wptr\_gray.sync\_wr\_ptr.\*.cdns\_syncflop\_inst.\*. u\_cdnsdru\_databasec\_synth.d\_in
  + rptr\_gray-> i\_cdnsdru\_databook c\_rptr\_gray.sync\_wr\_ptr。\*.cdns\_syncflop\_inst.\*. u\_cdnsdru\_databasec\_synth.d\_in
  1. Synthesis App笔记
  + 如果您使用Genus作为您的合成工具，请确保优化开关已打开。
  + 还要确保没有“模块保留”属性。
  + 如果您使用Synopsys DC作为合成工具，请确保“删除冗余逻辑”开关处于打开状态。
  + 请使用分层合成流程，不要使用扁平化合成流程。这将有助于减少LEC运行时间并适应智能LEC流程。
  + 确保您始终使用Genus的最新版本。

# 第12章：集成指南

第12章：**集成指南**

* 1. 启动序列

Cadence PCIe IP子系统具有不同的复位输入，这些复位输入被配置为与相应的时钟同步。复位输入可以异步置位，但应与相应时钟同步解除置位

控制器在内部将所有复位输入端连接到各自的时钟域。因此，客户端逻辑不需要在IP之外添加任何重置同步器。

以下各节提供有关复位输入的信息，以及它们应被激活的推荐顺序

* 1. 复位序列

控制器和PHY子系统的启动顺序如下。

 注：以下序列中复位解除断言的相对顺序很重要，建议步骤之间至少有1个时钟周期或更多延迟。

1. 接通PCIe子系统的电源，并断言PHY和控制器的所有重置，这：
   * 使能PHY的参考时钟
   * 启用IP*phy\_apb\_config*、*core\_apb\_config*、*pm\_config*和*axi\_config*的所有自由运行时钟。
2. 取消断言*POR\_N*（*上电复位*）和*PERST\_N*。

 注：这些信号是系统级复位信号，不存在于PCIe IP中。

1. 取消断言APB重置*链路 \*\_apb\_rst\_n*，以确保控制器的APB接口处于活动状态。
2. 取消断言APB重置*phy\_apb\_rst\_n*以确保PHY的APB接口处于活动状态。
3. 对物理寄存器进行编程并设置物理和控制器静态配置引脚。
4. 取消断言*phy\_rst\_n*复位输入以激活PHY。 注：这些是异步复位输入。

等待*pipe\_l\_phy\_status*从1转换为0，然后再进入下一步。

这表示控制器使用的PHY输出时钟*（core\_clock*和*pipe\_clock*）稳定。

 注：对于*phy\_p\*\_reset\_n*输入：

* 对于单链路配置，所有*phy\_p\*\_reset\_n*输入均应驱动为高电平。建议将SoC可编程寄存器连接为默认值1）
* 对于多链路配置，在取消断言*phy\_reset\_n*时：
  + 必须使能至少1个*phy\_p\*\_reset\_n*输入，即在*phy\_reset\_n*取消断言时使其*phy\_p\*\_reset\_n为高*。
  + 对于剩余的禁用链路（如果有的话），从时间0起或者在*PHY\_RESET\_N*的解除断言之前，应当将相应的PHY\_P\*\_RESET\_N驱动到低（1 ′ B0）。
  + 在初始PHY启动之后，在*phy\_reset\_n*的解除断言时被禁用的任何链路可以通过解除断言其*phy\_p\*\_reset\_n*（即，驱动它1'b1）来

1. 按照以下顺序取消断言控制器复位输入：a*core\_rst\_n*
2. axi\_rst\_n
3. reg\_sticky\_rst\_n
4. *\*\_reg\_sticky\_rst\_n*仅在上电后第一次进入时置位e取消置位*reg\_axi\_rst\_n*和*reg\_rst\_n*。

f取消断言*pm\_rst\_n*

S C P

l r t

C

R r

er e

是的

一个一 个

La L

L l

an

ck o

In i

It

g Lc

O a

L a s t

是啊

h D r

Pe T

咿呀咿呀

IC I

C PC

P P

PCIe串行引脚

物理管道rst n

PMA/PCS

phy\_abp\_rst\_n

管道初始值

下午

pm\_rst\_n

APB

apb\_rst\_n

核心rst\_n

注册号

reg\_sticky\_rst\_n

axi\_rst\_n

出境

应用界面

1. 如果在对ip管理寄存器进行编程后已置位，则等待*ip\_reg\_init\_in\_progress*解除置位
2. 按照“*编程IP寄存*器”部分设置控制器IP寄存器。
3. 断言*link\_training\_enable*输入以允许链接训练开始。



AXI\_CLK/apb\_CLK/

时钟 pm\_CHAPTER PHY\_REFCLK\_P\_N

link\*\_apb\_rst\_n phy\_apb\_rst\_n

APB重置 物理管道rst n

对于Core/PHY 物理状态

pma\*\_pll\_fullrt/pipe\_pull\_out

A（A）在进入下一步之前，对PHY寄存器进行编程并设置PHY/内核静态配置引脚

B

核心主重置

link\*\_pipe\_rst\_n

link\*\_core\_rst\_n

（B）等待phy\_status从1-> 0转换，然后进行下一步

AXI主复位

link\*\_axi\_rst\_n

link\*\_link\_sticky\_rst IP寄存器 \_n

重置 链接 \*\_rst\_n

电源管理 link\*\_pm\_rst\_n resetlink\*\_pm\_rst\_n

(C)\*\_reg\_sticky\_rst\_n仅在第一次进入后断言 C

上电

D

控制器 link\*\_link\_trainini列车启用 ng\_enable

（D）在进行下一步之前，如果需要，设置控制器寄存器

**图12.1：**复位顺序

12.2.1 复位参考框图

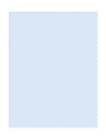
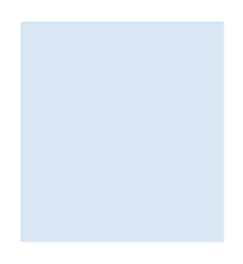
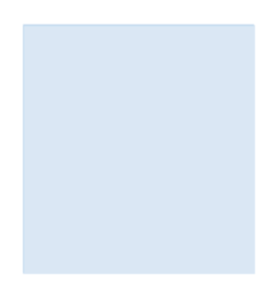
下图用于将复位集成到SoC内的PCIe IP（具有集成PHY的PCIe控制器），并连接到应用逻辑。实施重置序列的重置序列器在[*重置序列*](#_bookmark175)中进行了描述。复位同步器在控制器IP内部实现，以将复位同步到相应的时钟。

 注意：不需要在PCIe IP之外添加重置同步器。

D

4

I X A



客户PCIe SoC

Cadence PCIe IP

cmn\_ref\_png\_p cmn\_reg\_png\_m

B

C

:

:

:

:

x16车道

:

:

:

:

:

:

:

:

x16通道s

:

:

:

:

PERST\_N

上电复位

A.回收 B.管道安装 C. 堆芯 D. AXI时钟E. pm\_F. apb\_bits

时钟

**图12.2：**PCIe复位参考框图示例

AXI CLK

D

堆芯

C

apb\_bits

F

公司简介

E

管道安装

B

apb\_bits

F

管板

\_in

B

F

E

一

重置定序器

SoC参考时钟

入站

## 精密时间测量

系统重置

phy\_status（link0\_clock\_stable）

重置同步器

SoC复位逻辑

*精确时间测量*（PTM）通过独立的本地时钟实现跨多个组件的事件精确协调。一般来说，这种精确的协调将是困难的，因为单独的时钟

对时间的价值和变化率有不同的看法为了解决此限制，PTM使组件能够计算其本地时间与共享PTM主时间（与PTM根关联的独立时域）之间的关系。

* + 1. 特征

控制器实现PTM的关键功能如下所示

* + - * PTM能力寄存器组在物理功能0中实现
      * 支持增强的精确时间管理。
      * 使用传播延迟自适应能力支持解释A和解释B
      * PTM调试器模式特点：
        + 配置为端点模式时，用作PTM调试器。
        + 实现基于core\_clock的本地时钟。
        + 为本地固件提供简单的寄存器编程机制，以便在设置PTM启用时触发PTM对话框
        + 在PTM对话框结束时计算当前PTM主机时间，并通过ptm\_master\_timer\_out[63：0]和ptm\_master\_timer\_out\_valid输出给用户
        + 可通过APB接口读取本地PTM上下文
      * PTM响应器模式功能：
        + 当配置为根端口模式时，用作PTM响应器。
        + 实现基于core\_clock的本地时钟。
        + 用户可通过ptm\_master\_timer\_in[63：0]和ptm\_master\_timer\_in\_valid输入预设本地时钟。
        + 设置PTM启用时，自动响应PTM请求
        + 可通过APB接口读取本地PTM上下文
    1. 功能描述
       1. PTM链路协议

下图显示了在控制器中实现的PTM机制

接收PTM请求和捕获时间戳t2

发送PTM响应和捕获时间戳t3

接收PTM请求并捕获时间戳t2'

PTM应答者

发送具有（t2 '，t3-t2）和捕获时间戳t3'的

t2 t3 t2't3'

PTM请求

PTM

对话1

PTMPTM请求

响应

PTM

对话2

PTM

收到答复D

客户端触发器

t1 t4 t1't4'PTM转换器

*LCRC、ECRC*

检查

输出

时间

发送PTM请求和捕获时间戳t1

接收PTM响应并捕获时间戳t4

发送PTM请求和捕获时间戳t1'

*PTM\_local\_timer\_out*

接收PTM响应和捕获

时间戳t4 '，t2'，（t3-t2）

**图12.3：**PTM链路协议

计算t1'时的PTM主

PTM滤波器

在端点模式下运行时，控制器用作PTM调试器在此模式下，控制器向客户端提供触发PTM对话进程的机制。客户端可通过APB接口对本地“PTM控制寄存器”进行编程，从而触发PTM对话框。

在该触发事件时，控制器通过PCIe链路发送PTM请求消息当PTM请求消息报头（STP令牌）被发送时，控制器捕获

控制器等待接收PTM响应/PTM响应D消息。当接收到PTM响应/响应D消息时，控制器捕获时间控制器在注册捕获的时间戳之前执行LCRC、ECRC、TLP报头检查。

当接收到PTM响应D消息时，控制器从消息中提取PTM响应者时间戳（t2'和

使用该信息，控制器使用以下表达式计算t1'时的PTM主时间

* \* （在t1 '处的PTM主时间）= t2'-（（t4-t1）-（t3-t2））/2
  + 在响应D消息中接收的时间戳t1、t4、t1 '、t4'、传播延迟t3-t2存储在控制器的本地“PTM上下文寄存器”中，并且可由客户端通过APB接口读取。

此外，使用以下表达式计算当前PTM主时间

* \* （当前PTM主时间）=（t1 '时的PTM主时间）+（当前PTM本地定时器）-（t1'）
  + PTM\_MASTER\_TIMER\_OUT[63：0]输出“当前PTM主机时间”，PTM\_MASTER\_TIMER\_OUT\_VALID置位1个core\_OUT周期。

PTM应答者

在根端口模式下操作时，控制器用作PTM响应器在此模式下，控制器可编程为在收到PTM请求时自动响应PTM响应/响应D消息

当接收到PTM请求时，控制器捕获时间戳（t2/t2 '）。当PTM响应/响应D被发送时，控制器捕获时间戳（t3/t3 '）。从先前PTM对话存储的（t3-t2）值和捕获的t2'用于PTM ResponseD消息。

时间戳t2、t3、t2 '、t3'存储在本地“PTM上下文寄存器”中，并且可由客户端通过APB接口读取

* + - 1. PTM时间戳捕获

控制器捕获的所有时间戳必须基于构成PTM消息TLP的STP符号或令牌，就像在端口引脚上观察该符号或令牌的第一位一样。

然而，实际上，在端口的引脚上捕获发送和接收串行数据的时间戳是不可行的控制器在Tx和Rx数据路径中的内部点捕获时间戳，然后使用延迟因子调整时间戳，直到端口的引脚。控制器用于PTM测量的内部数据路径如下图所示

APB

**接口**

PTM\_master\_timer\_out[63：0] PTM\_master\_timer\_out\_valid

目标接口

主接口

PTM\_master\_timer\_in[63：0] PTM\_master\_timer\_in\_valid

APB

**接口**

EP控制器（PTM控制器）

目标接口

**主接口**

RP控制器（PTM响应器）

AXI

AXI

PTM上下文本地寄存器

PTM

能力配置注册

LM/CFG/HAL

转换译码器

PTM

消息生成SM

转换译码器

PTM

消息生成SM

TL TL

Client\_Tx\_PNP\_Intfc

反式CRC

Client\_Tx\_PNP\_Intfc

反式CRC

友情链接

link\_Tx

友情链接

link\_Tx

DLL DLL

**PTM时间戳RX PTM时间戳RX**

64位定时器

64位定时器

帧解码器

帧编码器

帧解码器

帧编码器

PIPE Rx Intfc

PIPE Rx Intfc

PTM时间戳发送

PTM时间戳发送

PL PL

管道纠偏

管道纠偏

车道解扰器

车道扰频器

车道解扰器

车道扰频器

PIPE RX FIFO

PIPE TX FIFO

PIPE RX FIFO

PIPE TX FIFO

**e**

PCLK

管道\_

管

Interfac

时钟（\_C）

核心

e

**核心**

PCLK

管道\_

管

Interfac

时钟（\_C）

RP PHY

EP PHY

PHY

PHY

PTM请求

PTM应答D

PTM响应

**图12.4：**PTM实现的控制器框图

* + - 1. PTM时间戳

根据PCIe规范，时间戳必须基于构成TLP的STP符号或令牌，就像在端口引脚上观察该符号或令牌的第一位一样然而，这实际上并不可行，

控制器在数据路径的内部点捕获时间戳，如前所述。物理层使用静态可编程延迟值。然而，弹性缓冲区中的接收延迟可能存在动态变化。这在控制器和PHY设计中目前没有考虑因此，PTM测量值可能不准确，其程度如下表所示。

表12.1：PTM时间戳

|  |  |
| --- | --- |
| 时钟架构 | 最大不准确度 |
| Common Clock（0 ppm） | 0 ns（无弹性缓冲区变化。） |
| SRNS（600 ppm） | * 64 GT/s时= +/-0.5ns * 32 GT/s时= +/-1 ns * 16 GT/s时= +/- 2 ns * 8 GT/s时= +/- 4 ns * 5.0 GT/s时= +/- 8 ns * 在2.5 GT/s = +/- 16 ns |
| SRIS（5600 ppm） | * 64 GT/s时= +/-1 ns * 32 GT/s时= +/-2ns * 16 GT/s = +/- 4 ns * 8 GT/s时= +/- 8 ns * 5.0 GT/s时= +/- 16 ns * 2.5 GT/s = +/- 32 ns |

确保最佳精度

建议使用以下系统设置，以最大限度地减少延迟变化并确保最大精度：

* + - * + 禁用ASPM L0、L1

在PTM对话框之间进入L0、L1可能会导致控制器中的Rx偏斜和接收FIFO电平发生变化

* + - * + Link以16 GT/s的速度

这可确保器件以最高频率运行，并尽可能减小PTM测量值的变化。

* + - * + 使用通用时钟

如果可行，使用通用时钟架构可确保接收弹性缓冲器中的0 ppm和0变化

* + - 1. 异常处理
         * 不支持的请求

当控制器配置为下游端口时，如果在PTM启用位清零时收到PTM消息，则该消息将被视为不支持的请求。

此错误在AER和设备状态寄存器中报告为不支持的请求。

PTM消息不会转发到客户端接口。

当控制器被配置为上游端口时，如果在PTM启用位被清除时接收到正确形成的PTM响应消息，则该消息被静默丢弃。

PTM消息不会转发到客户端接口。

* + - 1. PTM上下文无效

PCIe规范，强烈建议当PTM主时间和上游端口的本地时间之间的关系改变时，上游端口使其内部PTM上下文无效，如由实现特定的标准所确定的例如，这可能由于转换到非D0状态或由于累积的PPM漂移而发生。

控制器在端点模式下实现两种PTM上下文无效模式

* + - * + 客户端使PTM上下文无效：

客户端固件可以通过向“PTM本地控制寄存器”的“客户端无效PTM上下文”位27写入“1”来使PTM上下文

客户端可以通过设置“PTM本地控制寄存器”中的PTM触发器启用位来启用PTM对话框，以获取当前PTM上下文和PTM主时间。

* + - * + 硬件自动失效PTM上下文：

当LTSSM转换出L0状态时，控制器可以自动使PTM上下文

默认情况下启用此模式可通过编程“PTM本地控制寄存器”的位28“禁用自动无效PTM上下文”来禁用此模式

当PTM上下文自动失效时，控制器通过断言local\_interrupt输出来通知客户端。“PTM上下文自动无效”状态位-27也在“本地错误状态寄存器0”中设置

* 1. SRAM集成

与内核使用的RAM相关的接口信号在设计的顶层可用，以便轻松与客户生成的RAM模块接口。这也使用户能够在不修改任何Cadence PCIe核心设计文件的情况下向RAM添加BIST功能

所需SRAM的详细信息，包括时钟关系和读取延迟，请参见docs/cdnpcie\_sram\_requirements.html文档

控制器发出的读和写访问描述如下：

RAM读取

* + - * + 控制器驱动读取地址和读取使能1个时钟，并在2个周期后对读取数据进行采样，如图所示

在配置时，还可以选择3周期SRAM访问延迟

* + - * + 控制器从不对未初始化的RAM位置发出读访问
        + 控制器可以发出连续的背靠背RAM读取访问。
        + 写入RAM位置后，控制器可以在写入后1个周期向同一位置发出读取
        + 控制器具有内部机制，以防止读取和写入访问冲突。
        + 控制器从不同时对同一RAM位置发出读和写即使读写时钟是异步的，控制器也能确保这一点

时钟



1

2

3

4

5

6

7

8

9

10

11

地址1 地址2

Data1 Data2

地址3地址4

数据3数据4

写使能

写地址

写入数据

读地址

读取使能

读取数据

RAM写入

* + - * + 控制器驱动器写地址，写数据和写使能为1个时钟周期。
        + 控制器可以发出连续的背靠背RAM写访问。
        + 读取RAM位置后，控制器可以在读取后1个周期向同一位置发出写入
        + 控制器具有内部机制，以防止读取和写入访问冲突。
        + 控制器从不同时对同一RAM位置发出读和写即使读写时钟是异步的，控制器也能确保这一点

**集成说明：**控制器可以在写入后立即发出读取，或者在读取后立即向同一位置发出写入。因此，在集成过程中，确保读和写权限相同非常重要。作为示例实现，集成器可以使用标准的1周期RAM宏，并在所有RAM输入上添加寄存器片，即（地址/写数据/读使能/写使能）。在RAM实现中必须保持先写后读的顺序

* 1. 重要产品数据（VPD）集成

*重要产品数据*（VPD）是唯一标识系统的硬件和潜在软件元素的信息VPD可以向系统提供关于各种现场可更换单元的信息，例如部件号、序列号和其他详细信息。从系统的角度来看，目标是使系统所有者和服务人员可以获得这些信息

VPD驻留在PCI设备中的存储设备中，例如串行EEPROM使用配置空间中的功能列表提供对VPD的访问

此VPD功能寄存器按物理功能实现，并存在于端点中

请参阅交付包的*<package\_path>/models/ipxact/\**目录中的寄存器描述中的VPD功能寄存器这是一个可配置的功能。只有在IP配置器中选择时，此功能和寄存器才可用。

下面的例子展示了如何使用它。

* + 1. VPD读取示例

1. 尝试读取VPD的主机将向VPD功能结构发送包含以下信息的CFG写入
   * VPD标志=零，VPD地址=需要由主机读取的VPD地址
2. 控制器IP在看到对VPD能力结构的VPD地址的写入时将生成VPD\_INTERRUPT（*cap\_state\_pf\_out*信号的一部分）单周期脉冲，该单周期脉冲向用户应用指示：

VPD能力有新数据。

* + PF 0组上的脉冲表示PF 0的VPD事件，PF 1组上的脉冲表示PF 1的VPD事件，依此类推。

1. 在看到“VPD\_INTERRUPT”时，在端点上运行的本地软件必须读取相应功能的VPD能力寄存器，以确定主机正在请求读/写。
   * “VPD Flag”0表示主机请求读取
2. 本地软件应使用ip-mgmt接口读取VPD标志、VPD地址
3. 一旦本地软件具有用于VPD读取请求的数据，它就应该使用ip-mgmt接口将数据写入VPD能力结构中的VPD数据寄存器，并将VPD标志设置为1。
4. 主机软件将监控VPD标志，一旦设置为1，将使用VPD数据寄存器中的
   * 1. VPD写入示例
5. 尝试写入VPD的主机将向VPD功能结构发送包含以下信息的CFG写入
   1. VPD数据=要写入的
   2. VPD标志= ONE，VPD地址=需要写入的VPD地址
6. 控制器IP在看到对VPD能力结构的VPD地址的写入时将生成VPD\_INTERRUPT（cap*\_state\_pf\_out*信号的一部分）单周期脉冲，其向用户应用指示VPD能力具有新数据。

- PF 0组上的脉冲指示PF 0的这种VPD事件，PF 1组上的脉冲指示PF 1的VPD事件，等等。

1. 在看到VPD\_INTERRUPT时，在端点上运行的本地软件必须读取相应功能的VPD能力寄存器，以确定主机正在请求读/写。

- VPD标志ONE指示主机请求写入

1. 本地软件应使用ip-mgmt接口读取VPD标志、VPD地址和VPD数据
2. 本地软件写入VPD写入请求的数据后，应使用ip-mgmt接口将VPD标志设置为零
3. 主机软件将监视VPD标志，并且一旦将其设置为零，则现在将知道VPD写入请求现在已完成。
   1. 热插拔集成
      1. 概述

PCI Express定义了用于对下游端口的热插拔支持的机制，并提供了允许使用自一致的基础设施来支持不同用户/操作员模型此功能适用于根联合体和交换机下游端口。

[*表12.2*](#_bookmark188)中描述了PCIe规范中定义的热插拔元素。

表12.2：热插拔要素

|  |  |
| --- | --- |
| 元件 | 目的 |
| 指标 | 显示插槽 |
| 手动操作固定闩锁（MRL） | 将适配器固定到位 |
| MRL传感器 | 允许端口和系统软件检测正在打开 |
| 机电联锁 | 防止适配器从插槽中 |
| 注意按钮 | 允许用户请求热插拔操作 |
| 软件用户界面 | 允许用户请求热插拔操作 |

表12.2：热插拔

|  |  |
| --- | --- |
| 元件 | 目的 |
| 插槽编号 | 提供插槽 |
| 功率控制器 | 软件控制的电子组件，用于控制插槽或适配器的电源，并监控电源的故障状况 |

链路

热插拔PCIe交换机

链路

热插拔信号

链路

热插拔信号

时隙1

插槽2

热插拔端口2

热插拔端口1

上游端口

支持热插拔的控制器

支持热插拔的控制器

控制器

**图12.5：**支持热插拔的PCIe交换机

需要若干接口信号来支持如PCIe规范所描述的热插拔元件将控制器连接到热插拔插槽所需的接口信号如[*图12.6*](#_bookmark189)所示。

|  |
| --- |
| 插槽容量寄存器 |
| 插槽控制寄存器 |
| 插槽状态寄存器 |

**图12.6：**热插拔接口方框图

热插拔

通过USB的热插拔事件MRL\_CN\_N的软件通知

PRSNT\_N

注意按钮N

PCIe控制器（下游端口）

电源故障N

EMI\_状态

ATTN\_INDICATOR[1：0]

PWR\_INDICATOR[1：0]

PWR\_CTRL

EMI\_CTRL

命令\_已更改

命令\_已完成

PCIe链路

槽

* + 1. 信号描述

[*表12.3*](#_bookmark191)显示了与热插拔实现相关的信号

表12.3：热插拔接口输入和输出

|  |  |  |  |
| --- | --- | --- | --- |
| 信号 | 方向 | 宽度 | 描述 |
| 热插拔mrl传感器n | 输入 | 1 | 指示手动固定闩锁（MRL）的状态。MRL传感器是一种开关、光学设备或其他类型的传感器，可将插槽的MRL位置报告给  下游港口。当MRL完全关闭和打开  在所有其他时间（即，如果MRL完全打开或处于中间位置）。  如果MRL完全关闭=> 0在所有其他时间=> 1 |
| 热插拔 | 输入 | 1 | 低电平有效输入，表示插槽中存在适配器。 |

表12.3：热插拔接口输入和输出

|  |  |  |  |
| --- | --- | --- | --- |
| 信号 | 方向 | 宽度 | 描述 |
| 热插拔注意按钮 | 输入 | 1 | 当按下由底盘电控的注意按钮  0 =>按下  1 =>默认状态 |
| 热插拔电源故障 | 输入 | 1 | 电源故障指示灯产生的外部电源故障检测，灰逻辑。低电平有效 |
| 热插拔emi状态 | 输入 | 1 | 机电联锁状态输入。 |
| 热插拔命令已更改 | 输出 | 1 | 1-位输出，以指示发生了对插槽控制寄存器的写入，这导致热插拔命令输出信号之一发生变化。  这将被设置为高，直到COMMAND\_COMPLETED在-  put被设置为“1”。  如果“没有命令完成支持”=1插槽功能寄存器，此输出可以忽略，nored，将被绑低。 |
| 热插拔命令完成 | 输入 | 1 | 命令已完成状态输入以响应COMMAND\_CHANGED。  高电平有效输入，置位1个时钟周期。  如果插槽功能寄存器中的“No Command Completed Support”=1，则此输入可连接到“0”。 |
| 热插拔中断输出测试 | 输出 | 1 | 热插拔事件的软件通知的输出 |
| 热插拔指示器 | 输出 | 2 | 注意指示器控制输出.定义的编码为：00b：保留  01 b：- On - Attention -此插槽  10 b：-闪烁-定位-根据用户请求识别  11b：-关闭-正常-正常运行 |

表12.3：热插拔接口输入和输出

|  |  |  |  |
| --- | --- | --- | --- |
| 信号 | 方向 | 宽度 | 描述 |
| 热插拔指示器 | 输出 | 2 | 电源指示灯控制输出：定义的编码为：  00b：保留  01 b：On - Power On -不允许插入或移除适配器。  10 b：闪烁-电源转换-热插拔操作正在进行，插入或重新插入  不允许移动适配器。  11b：关闭-电源关闭-允许插入或移除适配器。 |
| 热插接式断路器 | 输出 | 1 | 根据定义的编码设置插槽的电源状态  定义的编码为：0b：Power On  1b：关闭 |
| 热插拔EMI电容器 | 输出 | 1 | 机电联锁控制输出。  定义的编码为：  0b：机电互锁分离  1b：机电互锁接合 |

* + 1. Cadence PCIe控制器热插拔实施

[*图12.7*](#_bookmark193)显示了控制器中热插拔功能的实现

|  |  |  |  |
| --- | --- | --- | --- |
|  | 数据同步 |  | 下降  边缘检测 |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | 数据同步 |  | 下降  边缘检测 |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | 数据同步 |  | 变化  检测逻辑 |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | 数据同步 |  | 变化  检测逻辑 |
|  |

**图12.7：**控制器热插拔实现



插槽状态寄存器

插槽控制寄存器

按下注意按钮启用

注意按钮

按下EN

注意按钮N

St

内核\_时钟

检测到电源故障

电源故障检测使能

EN

热插拔

电源故障N

St

REG

内核\_时钟

MRL已更改

MRL已更改启用

EN

MRL\_N

St

内核\_时钟

存在检测已更改

存在检测已更改启用

EN

PRSNT\_N

St

内核\_时钟

命令已完成

命令完成中断使能

EN

**命令\_已完成** 数据

同步

St

内核\_时钟

DLL状态已更改启用

DLL状态已更改启用

EN

从链接状态

寄存器

DLL活动位

St

热插拔中断启用

EN

EMI\_状态

EMI\_状态

ATTN\_INDICATOR[1：0]

数据同步

St

收件人指示器

内核\_时钟

PWR\_INDICATOR[1：0]

PWR指示器

PWR CTRL

PWR\_CTRL

切换EMI CTRL

EMI\_CTRL

命令\_已更改

命令更改检测

* + - 1. 插槽状态寄存器实现

具有热插拔功能的下游端口支持以下热插拔事件：

* + - * + 插槽事件：

**按下注意按钮**- 由hot\_plug\_attention\_button\_n输入指示

**检测到电源故障**- 由hot\_plug\_power\_fault\_n输入指示

**MRL传感器已更改**- 由hot\_plug\_mrl\_sensor\_n输入指示该输入的变化由控制器检测并记录在状态寄存器中。

**存在检测已更改**- 由hot\_plug\_prsnt\_n输入指示该输入的变化由控制器检测并记录在状态寄存器中。

* + - * + 命令完成事件

由hot\_plug\_command\_completed输入指示。

* + - * + 数据链路层状态更改事件

由“链路状态寄存器”DLL活动位指示。

这些事件中的每一个都具有状态字段和启用字段，状态字段指示事件已经发生但尚未被软件处理，启用字段指示事件是否启用软件通知。某些事件还有一个capability字段，用于指示端口是否支持该事件类型。

* + - 1. 插槽控制寄存器实现

使能位用于使能各种热插拔事件的中断生成如[*图12.7*](#_bookmark193)所示，控制输出用于将输出直接驱动到插槽。

* + - 1. 热插拔事件

热插拔事件通过hot\_plug\_interrupt\_out中断输出通知。此输出将保持置位，直到通过向插槽状态寄存器中的位写入“1”来清除状态位。

* + 1. 寄存器描述

热插拔能力结构仅适用于下游端口。

* + - 1. 插槽功能寄存器地址：0xD4

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 物理插槽编号[12：0] [31：19] | 18 | 17 | 十六点十五分 | 插槽功率限制值[7：0]  [14：7] | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

无命令完成支持

机电联锁当前时隙功率极限标度

热插拔功能热插拔惊喜

电源指示灯存在注意指示灯存在MRL传感器存在

电源控制器存在注意按钮存在

|  |  |  |  |
| --- | --- | --- | --- |
| 位 | 寄存器描述 | 属性 | 执行 |
| 0 | **注意按钮存在**-设置时，此位表示此插槽的注意按钮由机箱电气控制 | HwInit | 没有逻辑。  来自APB接口的 |
| 1 | **电源控制器存在**-设置时，此位表示已为此插槽/适配器实现软件可编程电源控制器（取决于外形尺寸）。 | HwInit | 没有逻辑。  来自APB接口的 |
| 2 | **MRL传感器存在**-设置时，此位表示机箱上已为此插槽安装了MRL传感器。 | HwInit | 没有逻辑。  来自APB接口的 |
| 3 | **注意指示灯存在**-设置时，此位表示注意指示灯由底盘电气控制。 | HwInit | 没有逻辑。  来自APB接口的 |
| 4 | **电源指示灯存在**-设置时，此位表示此插槽的电源指示灯由机箱进行电气控制。 | HwInit | 没有逻辑。  来自APB接口的 |

|  |  |  |  |
| --- | --- | --- | --- |
| 位 | 寄存器描述 | 属性 | 执行 |
| 5 | **热插拔意外**-设置时，此位表示此插槽中的适配器可能会从系统中  没有任何事先通知。这是一种特定于外形的能力。此位指示操作系统允许在不影响软件继续运行的情况下进行此类删除。 | HwInit | 没有逻辑。  来自APB接口的 |
| 6 | **Hot-Plug Capable**-设置时，此位表示此插槽能够支持热插拔操作。 | HwInit | 没有逻辑。  来自APB接口的 |
| 十四点七分 | **插槽功率限制值**-与插槽功率  限制比例值，指定由  插槽（参见第6.9节）或通过其他方式连接到适配器。  功率限制（以瓦特为单位）的计算方法是将此字段中的值乘以插槽  功率限制比例字段，除了  当时隙功率限制比例字段等于00b（1.0x）且时隙  功率限值超过EFh，则使用以下替代编码：  F0h = 250 W插槽功率限制F1h = 275 W插槽功率限制F2h = 300 W插槽功率限制  F3h至FFh =为高于时隙功率限值的值  300 W  如果插槽实现位为  集  写入此寄存器也会导致端口发送Set\_Slot\_Power\_Limit消息。  圣人  硬件/固件初始化之前的默认值为  0000 0000 b. | HwInit | 在根端口模式下，写入此寄存器会导致控制器发送Set\_Slot\_Power\_Limit消息。 |

|  |  |  |  |
| --- | --- | --- | --- |
| 位 | 寄存器描述 | 属性 | 执行 |
| 十六点十五分 | **插槽功率限制比例**-指定用于插槽  功率限值（见第6.9节）。  数值范围  00b = 1.0x  01b = 0.1x  10b = 0.01x  11b = 0.001x  此寄存器必须实施，如果实施插槽位设置。  写入此寄存器也会导致端口发送Set\_Slot\_Power\_Limit消息。  硬件/固件初始化之前的默认值为00 b。 | HwInit | 在根端口模式下，写入此寄存器会导致控制器发送Set\_Slot\_Power\_Limit消息。 |
| 17 | **机电联锁存在**-设置时，此位表示机电联锁  机械联锁是简单的，  关于  此插槽的机箱。 | HwInit | 没有逻辑。  来自APB接口的 |
| 18 | **无命令完成支持**-设置时，此位指示此插槽不生成软件通知  当发出的命令由热插拔控制器完成时。只有在热插拔  capable端口能够接受对插槽控制寄存器所有字段的写入，而不会在连续写入之间产生延迟。 | HwInit | 没有逻辑。  来自APB接口的 |
| 三十一点十九分 | **Physical Slot Number（物理插槽号）**-此字段指示连接到此端口的物理插槽号。此字段必须在硬件上初始化为一个值，该值分配的插槽号在机箱中是唯一的，而不考虑与插槽关联的外形尺寸。对于连接  连接到集成在系统主板上或与交换机设备或根设备集成在同一芯片中  端口 | HwInit | 没有逻辑。  来自APB接口的 |

* + - 1. 插槽控制寄存器地址：0xD 8

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 十五点十三分 | 12 | 11 | 10 | 九点八分 | 七点六分 | 5 | 4 | 3 | 2 | 1 | 0 |

RsvdP数据链路层状态更改启用

机电联锁控制

电源控制器控制电源指示灯控制

注意指示灯控制热插拔启用

命令完成启用状态检测更改启用

MRL传感器改变启用检测到电源故障启用注意按钮按下启用

表12.4：插槽控制寄存器描述

|  |  |  |  |
| --- | --- | --- | --- |
| 位 | 寄存器描述 | 属性 | 执行 |
| 0 | **按下注意按钮启用**-当设置为1b时，此位启用按下注意按钮时  事件（见第6.7.3节）。  如果插槽功能寄存器中的注意按钮存在位为0b，则允许该位为只读，值为0b。  该位的默认值为0b。 | RW | **与按下注意按钮状态**进行AND以生成中断 |
| 1 | 电源故障检测使能  - 设置时，该位使能电源故障事件的软件通知（参见第6.7.3节）。  如果支持电源故障检测的电源控制器不  实现时，允许此位为只读，值为  0b.  该位的默认值为0b。 | RW | 与**电源故障检测状态**进行AND运算，生成中断 |

表12.4：插槽控制寄存器描述

|  |  |  |  |
| --- | --- | --- | --- |
| 位 | 寄存器描述 | 属性 | 执行 |
| 2 | MRL传感器更改启用  - 设置时，此位启用MRL传感器更改事件的软件通知（参见  第6.7.3节）。  如果插槽功能寄存器中的MRL传感器存在位为清除，则允许将  只读，值为0b。  该位的默认值为0b。 | RW | 与**MRL传感器更改状态进行AND**运算，以生成数据表 |
| 3 | **Presence Detect Changed En- able**-设置时，此位启用软件通知  存在检测已更改  事件（见第6.7.3节）。  如果插槽功能寄存器中的热插拔功能位为0b，  允许该位是只读的，其值为0B。  该位的默认值为0b。 | RW | 与**存在检测更改状态**进行AND运算以生成 |
| 4 | **Command Completed Interrupt Enable**-如果Command Completed（命令已完成）通知已启用，  移植（如果无命令  插槽功能寄存器中的完成支持位为0 b），当设置时，此位启用软  当一个热-  插头  命令由热插拔控制器完成。  如果不支持命令完成通知，则此位必须硬连线至0b。  该位的默认值为0b。 | RW | 与**命令完成状态进行AND**运算以生成中断 |
| 5 | **热插拔中断使能**-设置时，此位使能在以下情况下生成中断：  启用热插拔事件。  如果插槽功能寄存器中的热插拔功能位为清除，则允许将  只读，值为0b。  该位的默认值为0b。 | RW | 主机使能所有热插拔事件中断生成 |

表12.4：插槽控制寄存器描述

|  |  |  |  |
| --- | --- | --- | --- |
| 位 | 寄存器描述 | 属性 | 执行 |
| 七点六分 | **注意力指示器控制**- 如果实施了注意力指示器，则写入此字段  将注意力指示器设置为  书面状态。  此字段的读取必须反映来自最新写入的值，即使相应的热-  plug命令未完成，  除非软件发出写操作而不等待（如果需要），  完成前一个命令，在这种情况下，读取值  未定义。  定义的编码为：00b保留  01b开10b闪烁11b关  注意：此字段的默认值必须是非保留值之一。如果检察官-  中的tion指示器存在位  插槽功能寄存器为0b，此位允许为只读  值为00B。 | RW | 驱动器ATTN\_INDICATOR[1：0]  输出 |

表12.4：插槽控制寄存器描述

|  |  |  |  |
| --- | --- | --- | --- |
| 位 | 寄存器描述 | 属性 | 执行 |
| 九点八分 | **电源指示灯控制**-如果实施电源指示灯，则写入此字段以设置电源  指示器变为写入状态。  此字段的读取必须反映来自最新写入的值，即使相应的热-  plug命令未完成，  除非软件发出写操作而不等待（如果需要），  完成前一个命令，在这种情况下，读取值  未定义。  定义的编码为：00b保留  01b开10b闪烁11b关  注意：此字段的默认值必须是非保留值之一。如果功率  插槽中的指示器存在位  能力寄存器为0b，则允许该位为只读，值为00b。 | RW | 驱动PWR\_INDICATOR[1：0]  输出 |

表12.4：插槽控制寄存器描述

|  |  |  |  |
| --- | --- | --- | --- |
| 位 | 寄存器描述 | 属性 | 执行 |
| 10 | **功率控制器控制**-如果实现了功率控制器，写入时该位-  10设置电源状态  slot每个定义的编码。此位的读取必须反映最新写入的值，即使  如果相应的热插拔  命令未完成，除非软件发出写入（如果需要），否则不等待  之前的命令缓变到  在这种情况下，读取的值是未定义的。  注意，在某些情况下，功率控制器可以自主地  移除插槽电源或不响应基于  在检测到的故障条件下，取决于电源控制器控制设置。  定义的编码为：0b通电  1b关闭电源  如果插槽功能寄存器中的电源控制器存在位为清除，则写入  此位没有任何效果，  该位的读取值是不精确的。 | RW | 驱动PWR\_CTRL输出 |
| 11 | **机电联锁控制**-如果机电联锁  实现时，将1b写入  这  位导致互锁的状态切换。将0b写入此位无效。的读取  该位总是返回0b。 | RW | 驱动EMI\_CTRL输出 |

表12.4：插槽控制寄存器描述

|  |  |  |  |
| --- | --- | --- | --- |
| 位 | 寄存器描述 | 属性 | 执行 |
| 12 | **数据链路层状态更改启用**-如果数据链路  层链路活动报告能力为1b，此位在数据  链路层链路活动位为  变了  如果数据链路层链路主动报告功能位为0b，则允许读取该位。  值为0b。  该位的默认值为0b。 | RW | 与**DLL State Changed Status进行AND**运算以生成DLL |

* + - 1. 插槽状态寄存器地址：0xDA

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 十五点九分 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

RsvdP

数据链路层状态已更改机电联锁状态

存在检测状态MRL传感器状态命令已完成

存在检测更改MRL传感器更改电源故障检测注意按钮按下

表12.5：插槽状态寄存器描述

|  |  |  |  |
| --- | --- | --- | --- |
| 位 | 寄存器描述 | 属性 | 执行 |
| 0 | **按下注意按钮**-如果注意按钮简单-  当按下注意按钮  如果不支持注意按钮，则不得设置此位。 | RW1C | 由ATTENTION\_BUTTON\_N驱动-  放了。ATTENTION\_BUTTON\_N时，该位设置为“1”  从1到>0的变化。  将“1”写入此位时清0。 |

表12.5：插槽状态寄存器描述

|  |  |  |  |
| --- | --- | --- | --- |
| 位 | 寄存器描述 | 属性 | 执行 |
| 1 | **检测到电源故障**-如果实现了支持电源故障检测的电源控制器，则当电源控制器检测到  这个插槽出现了电源故障请注意，  取决于硬件能力，电源故障可以在任何时间被检测到，而与电源控制器控制设置或时隙占用无关。如果不支持电源故障检测，则该位不得设置。 | RW1C | 由POWER\_FAULT\_N输入驱动。当POWER\_FAULT\_N从1变为>0时，该位设为“1”。  将“1”写入此位时清0。 |
| 2 | **MRL传感器已更改**-如果实施MRL传感器，则当检测到MRL传感器状态更改  如果未安装MRL传感器，则此位不得置位。 | RW1C | 由MRL\_NOW\_N输入驱动。当MRL\_REQ\_N从1->0或0->1变化时，该位设置为“1”。  将“1”写入此位时清0。 |
| 3 | **存在检测已更改**-当存在检测状态位中报告的值更改时，该位置1。 | RW1C | 由PRSNT\_N输入驱动.当PRSNT \_N从1->0或0->1变化时，该位设置为“1”。  将“1”写入此位时清0。 |
| 4 | **命令完成**-如果命令完成通知是支持的（如果插槽功能寄存器中的无命令完成支持位为0 b），当热插拔命令完成且热插拔控制器准备接受子命令时，此位被设置。  命令已完成状态位被设置为指示主机软件热-  插头控制器已处理  前一个命令并准备好接收下一个命令;它不保证与该命令对应的动作完成。  如果不支持命令完成通知，则此位必须硬连线至0b。 | RW1C | 由COMMAND\_COMPLETED在-  放。此位在COMMAND\_COMPLETED  脉搏了  将“1”写入此位时清0。 |

表12.5：插槽状态寄存器描述

|  |  |  |  |
| --- | --- | --- | --- |
| 位 | 寄存器描述 | 属性 | 执行 |
| 5 | **MRL传感器状态**-此位报告MRL传感器的状态（如果实施）。  定义的编码为：0b MRL关闭  1b MRL打开 | RO | 连接至MRL\_CN\_N输入。 |
| 6 | **存在检测状态**-此位表示插槽中存在适配器，通过  物理的逻辑“或”  层带内存在检测机制，以及任何带外存在检测（  机制定义为  插槽的相应形状因子。注意，带内存在检测机制要求  将电源施加到适配器  因此，需要用于热插拔的功率控制器的形状因素必须实现物理引脚存在检测机制。  定义的编码为：0b插槽空  插槽中存在1b卡  此位必须在所有实现插槽的下游端口上实现。对于未连接到插槽的下游端口（其中PCI Express  能力寄存器为0b），此位必须硬连线至1b。 | RO | 连接到~PRSNT\_N输入。 |
| 7 | **机电联锁状态**-如果是机电联锁，  联锁已实现，此位指示机电联锁的状态。  定义的编码为：  0b机电互锁断开  1b机电联锁装置 | RO | 连接到EMI\_STATUS输入 |

表12.5：插槽状态寄存器描述

|  |  |  |  |
| --- | --- | --- | --- |
| 位 | 寄存器描述 | 属性 | 执行 |
| 8 | **数据链路层状态已更改**- 当  链路状态寄存器的数据链路层链路活动位中报告的值发生更改。  在响应一个数据链路层- er状态改变事件，软件必须读取数据链路层链路链路的链接状态寄存器位，以确定是否链接是活跃的，然后才启动，ING配置周期的热插拔设备。 | RO | 连接到链路状态寄存器DLL活动位 |

* 1. IP接口
     1. HLS PF/VF编码

PF/VF（ipfivf）编码用于HLS元数据和其他控制器接口。

**SR\_IOV**步幅在IP内部设置为1。在链路训练之前，可以使用本地管理寄存器动态ipfivf格式给出了附加PF中的VF编号和关联的PF编号。

PF也以相同的格式编码，其中一个比特指示编码是针对VF还是针对PF。该表示出了具有到PF的不同VF分配的ipfivf编码的示例。

表12.6：首字母缩略词和描述

|  |  |
| --- | --- |
| 缩写 | 描述 |
| 摆脱 | 与PCIe TLP |
| 国际石油天然气联合会 | 在关联PF |
| 总vf | 分配给PF的总VF，可通过LM寄存器编程。这是**SRIOV**只读功能字段。 |
| num\_vf | 由主机编程**的SRIOV**功能中的VF计数启用数量 |
| vf偏移 | **SRIOV**功能中的RO字段。（*vf\_offset*+ PF的RID）给出PF中第一个VF的RID。当*vf\_stride*不为1时，vf\_offset与vf\_stride相同当*vf\_Stride*为1时，*vf\_offset*为所有先前PF的（*num\_pfs*+（total\_vfs-1）） |
| vf\_Stride | **SRIOV**功能中的RO字段VF的*RID + vf\_Stride*给出PF中下一个VF的RID |



**图12.8：**颜色代码

* + - 1. 具有统一数量的总VF的示例

每个PF中有4个VF，主机仅启用了PF 0中的两个VF

|  |  |  |  |
| --- | --- | --- | --- |
| PF总数：2，VFS总数：每个PF 4，PF0.num\_vf=2，PF1.num\_vf=4：vf\_stride=1 | | | |
| 摆脱 | 国际石油天然气联合会 | | |
| BDF格式 | 相关PF | PF中的VF数量 | Is\_VF |
| 0 | 0 | - | 0 |
| 1 | 1 | - | 0 |
| 2 | 0 | 0 | 1 |
| 3 | 0 | 1 | 1 |
| 4 | 0 | 2 | 1 |
| 5 | 0 | 3 | 1 |
| 6 | 1 | 0 | 1 |
| 7 | 1 | 1 | 1 |
| 8 | 1 | 2 | 1 |
| 9 | 1 | 3 | 1 |

 注意事项：

在上表中，***突出显示***的条目具有禁用/未使用的VF实例。

* + - 1. 总VF数不均匀的示例

PF 0中有2个VF，PF 1中有6个VF可用主机在PF 1中仅启用了四个VF，在PF 0中仅启用了一个VF

|  |  |  |  |
| --- | --- | --- | --- |
| PF总数：2，Pf0.total VF = 2，PF1.total VF=6，PF0.num\_vf=1，PF1.num\_vf=4，步幅=1 | | | |
| 摆脱 | 国际石油天然气联合会 | | |
| BDF格式 | 相关PF | PF中的VF数量 | Is\_VF |
| 0 | 0 | - | 0 |
| 1 | 1 | - | 0 |
| 2 | 0 | 0 | 1 |
| 3 | 0 | 1 | 1 |
| 4 | 1 | 0 | 1 |
| 5 | 1 | 1 | 1 |
| 6 | 1 | 2 | 1 |
| 7 | 1 | 3 | 1 |
| 8 | 1 | 4 | 1 |
| 9 | 1 | 5 | 1 |

 注意事项：

* + - * + 在上表中，***突出显示***的行具有禁用/未使用的VF实例。
        + 在上表中，***带有 \****的行具有无效VF。
      1. HLS多VC处理

VC激活/停用和HLS接口

当VC的流控制握手完成时，该VC变为活动的可以通过清除VC能力寄存器中的VC使能来禁用VC当该VC中存在未决事务时，主机预计不会清除该VC。

控制器仅在任何一个相关VC激活时才激活入站HLS接口。在出站时，控制器仅在其中一个连接的VC变为活动时才响应激活请求在HLS变为活动状态后，控制器在以下情况下停用入站HLS总线：

* + - * + 客户端接口中的所有VC都处于非活动状态
        + *管道\_pipeline*为空
        + *deact\_hint*由客户端逻辑断言。

在出站HLS上，当与客户端接口相关联的所有VC变为不活动时，控制器断言deact\_hint。有一个IP寄存器可用于更改此行为。hal\_control\_hls字段选择是否在VC停用时停止

多VC集成详细信息

如果应用程序只需要一个VC，则不需要特殊的集成步骤

数据路径集成

如果配置只有一个客户端接口，则所有VC的TLP都按顺序传输到应用程序逻辑。应用程序逻辑不会阻塞特定VC的流量。如果应用程序用例期望每个VC都有畅通的路径，则需要选择多个客户端接口配置。

企业文化

如果多个VC映射到一个客户端接口，则应用程序逻辑必须在传输TLP之前检查可用VC信用这是为了防止该客户端接口中的行首阻塞。crd\_lmt\_to\_client\* 信号帮助在应用程序端实现信用检查逻辑。在应用程序端不实现此逻辑可能会导致临时阻塞，直到信用可用。

# 第13章：编程指南

第13章：编程指南

* 1. IP寄存器

在以下控制器复位解除置位后，可以通过AXI-Lite/APB接口对控制器IP寄存器进行

* + - 核心rst\_n
    - 注册号
    - reg\_sticky\_rst\_n
    - *apb\_rst\_n*。

 注意事项：

* + - * 在控制器复位被释放后，LTSSM将处于*检测安静*状态，并将启动链路训练过程
      * 所有寄存器编程必须在链路训练之前完成
      * 建议将控制器的*link\_training\_enable*输入保持为0，并完成**IP寄存器**

编程.

* + - * **IP寄存器**编程完成后，如果已置位，则等待*ip\_REg\_init\_in\_progress*解除置位
      * 只要*link\_training\_enable*输入为0，LTSSM将保持在*Detect.Quiet*状态
      * *link\_training\_enable*应该在编程之后被设置为1以允许链路训练开始。
  1. 软件驱动程序设置

软件驱动程序交付包包括以下文档，可参考驱动程序使用。

表13.1：文件可验证性

|  |  |
| --- | --- |
| 类别 | 项目详细信息 |
| 驱动程序移植指南 | 解释如何在不同平台上使用驱动程序 |
| 驱动程序快速入门指南 | 有关如何开始使用驱动程序的步骤，以及驱动程序文档的参考 |
| IP驱动程序指南 | 驱动程序的用户指南（使用Doxygen生成的API参考 |

* 1. 链路均衡设置
     1. **RP模式寄存器名称：**通道均衡控制寄存器。

（i\_PCIE\_sec\_ext\_cap\_struct/i\_lane\_equalization\_control\_reg\_[0，1，2，3，4，5，6，7]）

**寄存器偏移：**0x000030C、0x0000310、0x0000314、0x0000318、0x000031C、0x0000320、0x0000324、0x0000328

**寄存器写入数据：**0x0000000 \*（\* 检查写入数据的描述

**描述：**在RP模式下，通道均衡控制寄存器包含以下字段：

* + - * 下游端口8.0 GT/s变送器预设-默认值：1111 b
      * 下游端口8.0 GT/s接收器预设提示-默认值：111 b
      * 上游端口8.0 GT/s变送器预设-默认值：1111 b
      * 上游端口8.0 GT/s接收器预设提示-默认值：111 b

根据PCIe规范，这些字段的默认值为Invalid在链路培训之前，客户端应用程序固件必须

* + - * 8.0 GT/s TX预设的有效值为：0000 b至1010 b。
        + 8.0 GT/s RX预设提示的有效值为：000 b到110 b。

要编程的值是特定于应用的，需要由客户确定 注：如果此寄存器未在RP中初始化，则将训练链接到Gen3将不成功。

* + 1. RP模式

寄存器名称：PL 16GT通道均衡控制寄存器。（i\_regf\_pl\_16gts\_cap/ i\_pl\_16gts\_lane\_equalization\_control\_reg\_[0，1，2，3]）

**寄存器偏移：**0x00009E0、0x00009E4、0x00009E8、0x00009EC

**寄存器写数据：**0x 0000000 \*（\* 检查写数据的描述

**描述：**在RP模式下，PL 16 GT通道均衡控制寄存器包含以下字段：

* + - * + 下游端口16.0 GT/s变送器预设-默认值：1111 b
        + 上游端口16.0 GT/s变送器预设-默认值：1111 b

根据PCIe规范，这些字段的默认值为Invalid在链路培训之前，客户端应用程序固件必须将有效值编程

* + - * + 16.0 GT/s TX预设的有效值为：0000 b到1010 b。

要编程的值是特定于应用的，需要由客户确定 注：如果此寄存器未在RP中初始化，则将训练链接到Gen4将不成功。

* + 1. RP模式

寄存器名称：PL 32GT通道均衡控制寄存器。（i\_regf\_pl\_32gts\_cap/ i\_pl\_32gts\_lane\_equalization\_control\_reg\_[0，1，2，3]）

**寄存器偏移：**0x0000A60、0x0000A64、0x0000A68、0x0000A6C

**寄存器写数据：**0x 0000000 \*（\* 检查写数据的描述

**描述：**在RP模式下，PL 32GT通道均衡控制寄存器包含以下字段：

* + - * + 下游端口32.0 GT/s变送器预设-默认值：1111 b
        + 上游端口32.0 GT/s发射机预设-默认值：1111 b

根据PCIe规范，这些字段的默认值为Invalid在链路培训之前，客户端应用程序固件必须将有效值编程

* + - * + 32.0 GT/s TX预设的有效值为：0000 b到1010 b。

要编程的值是特定于应用的，需要由客户确定 注：如果此寄存器未在RP中初始化，则将训练链接到Gen5将不成功。

* + 1. EP模式

**寄存器名称：**Gen3链路均衡控制寄存器1。（i\_regf\_ip\_pl/i\_g3\_eq\_ctrl\_1）

**寄存器偏移：**0x1000478

**寄存器写入数据：**0x00000410

 注意：此设置需要与CDNS PHY一起操作。

**描述：**将“P4”编程为第三代均衡阶段2期间的预设请求

* + 1. EP模式

**寄存器名称：**Gen4链路均衡控制寄存器1。（i\_regf\_ip\_pl/i\_g4\_eq\_ctrl\_1）

**寄存器偏移：**0x100048C

**寄存器写入数据：**0x00000430

 注意：此设置需要与CDNS PHY一起操作。

**描述：**将“P4”编程为Gen4速度变更期间和Gen4均衡阶段2期间的预设请求

* + 1. EP模式

**寄存器名称：**Gen5链路均衡控制寄存器1。（i\_regf\_ip\_pl/i\_g5\_eq\_ctrl\_1）

**寄存器偏移：**0x10004A0

**寄存器写入数据：**0x00000630

 注意：此设置需要与CDNS PHY一起操作。

**描述：**将“P6”编程为Gen5速度变更期间和Gen5均衡阶段2期间的预设请求

* 1. 错误处理
     1. 接收端错误（入站）

表13.2：检测到的错误和相关操作

|  |  |  |
| --- | --- | --- |
| 错误说明 | Aer | 数据包/数据路径 |
| NPRT表RAM读取错误 | 带有奇偶校验错误指示的本地中断 | 需要对控制器进行 |
| 带有无效msg\_code的 | 在AER | TLP被丢弃  可以使用LM  寄存器hal\_control[0] |
| 在不支持功能时接收PTM。 | 在AER | TLP被丢弃 |
| 根端口中的PTM消息，PTM功能受支持但已禁用。 | 在AER | TLP被丢弃 |
| 终端中的PTM消息，支持PTM功能，但已禁用。 | 无声丢弃 | TLP被丢弃 |
| 根端口在禁用LTR时接收LTR | 在AER | 丢弃的TLP |
| 根端口接收OBFF | 在AER | 丢弃的TLP |
| 当OBFF使能不是01或10时， | 在AER | 丢弃的TLP |
| TLP针对重叠BAR | UR和LM寄存器指示标记了BAR重叠 | 丢弃的TLP |
| TLP未击中任何BAR | 在AER | 丢弃的TLP |
| WR TLP目标EROM | 在AER | 丢弃的TLP |
| MEM请求针对非D0状态 | 在AER | 丢弃的TLP |
| 带有非PASID或EXT\_STEPACK标签前缀的TLP | 在AER | 丢弃的TLP |
| 在pasid未启用 | 在AER | 丢弃的TLP |
| 在未启用扩展引导标记时具有扩展引导标记的TLP | 在AER | 丢弃的TLP |
| 中毒的原子 | 在AER | TLP丢弃。生成UR CPl。 |
| 带有数据的 | 在AER | 丢弃的TLP |

表13.2：检测到的错误和相关措施

|  |  |  |
| --- | --- | --- |
| 错误说明 | Aer | 数据包/数据路径 |
| 中毒非原子/VDM | 基于LM寄存器的建议错误/不可纠正  **tlp\_as\_advisory**设置 | TLP丢弃。 |
| 中毒TLP，无数据 | 对于没有数据的 |  |
| 请求进行FLR的 | 没有错误 | TLP静默丢弃 |
| 带有ID和EP路由的MSG，收到的总线号和设备号不正确 | 没有错误  在这种情况下，不检查总线和设备编号 | TLP未丢弃。 |
| 带有ID和EP路由的消息，收到时带有未实现的功能号 | 在AER | 丢弃的TLP |
| 在端点中未启用PF/VF时请求目标MEM或IO | 在AER | 丢弃的TLP |
| 以非D0 | 意外完成错误 | TLP丢弃。将此错误检测为非预期完成可通过LM寄存器*hal\_control[2]* |
| 针对FLR下 | 基于LM寄存器*hal\_conrol的*意外完成错误/静默 | TLP丢弃。LM寄存器*hal\_control[3]* |
| 标记无效的CPL | 在AER中标记意外完成错误 | TLP丢弃 |
| 请求者ID不匹配 | 在AER中标记意外完成错误 | 传输到客户端接口，具有完整的位和错误代码集。标签信息被丢弃。 |
| 中毒完成 | 在AER | TLP数据包被丢弃。标签信息也被丢弃 |
| 完成时间 | 完成超时在AER请求报头的较低地址中报告，标记、ATR、TC和类型字段被记录 | 在HLS\_IB处生成完成TLP，并在Metada- ta |
| 内存请求完成与字节计数，低地址问题。 | 在AER中标记意外完成错误 | 默认情况下，TLP将被丢弃。可以使用**ip\_management**寄存器将此TP传输到HLS接口，并在元数据中设置适当的*error\_code* |

在入站路径上，可以使用IP寄存器更改以下错误默认丢弃行为

hal\_cancel\_decoded\_tlp\_filter：

* + - * 中毒性咨询错误
      * 中毒不可纠正错误
      * 不支持的已过帐
      * 由于某些条件而被标记为静默丢弃的数据包。
      * 带有无效标记的
      * 使用有效标记的
      * 中毒的完成。

以下错误被标记为函数非特定错误并记录在所有PF中：

* + - * TLP不匹配任何BAR
      * RID字段不匹配的完成。

### 表13.3：格式错误

|  |  |  |
| --- | --- | --- |
| 错误说明 | Aer | 信用更新 |
| 选择ATS且AT=1 MEM Rd的长度=1或大于RCB设置 | 畸形错误 | 数据包被丢弃而不写入接收缓冲区。信用不退还。 |
| RRS（CRS/MRS）非CFG和DMWR请求 | 畸形错误 | 将为此错误返回积分默认情况下，TLP将被 |
| 具有匹配ID但意外TC、Attr或TC或字节计数或低地址的 | 畸形错误 | 将为此错误返回积分默认情况下，TLP将被 |
| 来自请求的ID匹配但TC不匹配的内存完成 | 畸形错误 | 将为此错误返回积分默认情况下，TLP将被 |

* + 1. HLS错误

表13 - 4：HLS出站错误

|  |  |  |
| --- | --- | --- |
| 错误说明 | HLS的行动 | 数据包丢失（是/否） |
| OB完成，CA状态 | 来自元数据的NP请求标头记录在AER | 没有 |
| 设置了中毒位的TLP | 中毒的TLP发送到PCIe链路 | 没有 |
| 信用类型不正确的TLP进入客户端界面 | 在LM寄存器**aue\_status** | 否。客户端逻辑不会创建此错误。这可能会导致系统挂起。 |
| 属于不同客户端接口的TC设置错误的TLP | 在LM寄存器**aue\_status** | 否。客户端逻辑不会创建此错误。这可能会导致系统挂起。 |
| TLP大小大于最大有效负载大小/最大读取请求大小 | 在LM寄存器**aue\_status** | 否。客户端逻辑不会创建此错误。这可能会导致系统挂起。 |
| PCIe标签值超出支持范围 | 在LM寄存器**aue\_status** | 否。客户端逻辑不会创建此错误。这可能会导致系统挂起。 |
| PCIe标记值已在使用 | 在LM寄存器**aue\_status** | 否。客户端逻辑不会创建此错误。这可能会导致系统挂起。 |

# 第14章 调试指南

第14章 调试指南

* 1. 链接训练
     1. 链接培训状态寄存器

要检查链路训练参数的状态，可以读取以下寄存器

* + - 1. 物理层故障状态寄存器0 i\_regf\_ip\_pl/i\_dbg\_sts\_0，寄存器偏移：0x 1000420包括物理层功能状态。

表14.1：物理层寄存器0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 位  位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 0 | 链路状态 | RO | 链路1的当前状态=链路训练完成，  0=链路训练未完成 | 0 |
| 三比一 | 保留 |  |  |  |
| 六比四 | 协商车道数 | RO | 在链路训练期间与另一方协商车道数  000 = x1，  001 = x2，  010 = x4，  011 = x8  100 = x16 | 0 |
| 九点七分 | 协商速度 | RO | 链路当前运行速度  000 = 2.5G，  001 = 5G，  010 = 8G，  011 = 16 G  100= 32 G | 0 |
| 十一点十分 | 保留 |  |  |  |
| 十九点十二分 | 接收的链接ID | RO | 链路训练期间从另一端收到的链路ID | 0 |
| 二十七点二十分 | LTSSM状态 | RO | LTSSM的当前状态。状态的编码在Ap-pennsylvaniaC中给出。 | 0 |

表14.1：物理层寄存器0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 位  位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 28 | 远程链路宽度向上-配置能力状态 | RO | 此字段中的1表示远程  节点在训练  中的序列  配置。链路启动时的完整状态。0表示远程节点未设置链路升级配置位。 | 0 |
| 三十一点二十九分 | 保留 |  |  |  |

* + - 1. 物理层故障状态寄存器1 i\_regf\_ip\_pl/i\_dbg\_sts\_1，寄存器偏移：0x 1000424包括物理层功能状态。

表14.2：物理层UART状态寄存器1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 位  位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 十五比零 | 协商车道地图 | RO | 该字段的第i位在链接  如果通道i是PCIe链路的一部分，则进行训练。此字段的值仅有效  当链路处于L0或  L0s国家。 | 0 |
| 16 | 车道恢复状态 | RO | 如果LTSSM具有以下特性，则  反车道行驶  数字形成联系。 | 0 |
| 三十一点十七分 | 保留 |  |  |  |

* + - 1. 物理层缓存状态寄存器2 i\_regf\_ip\_pl/i\_dbg\_sts\_2，寄存器偏移：0x 1000428接收FTS计数状态寄存器。

表14.3：物理层UART状态寄存器2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特位置 | 寄存器字段 | Attr | 描述 | 复位 |

表14.3：物理层UART状态寄存器2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 七比零 | Gen1的接收FTS计数 | RO | 期间从另一侧收到的FTS计数  链接培训，  2.5 GT/s链接速度。控制器传输这么多FTS序列，  退出时的序列  当以2.5 GT/s的速度运行时，L0 S状态。 | 0x0 |
| 十五点八分 | Gen2的接收FTS计数 | RO | 期间从另一侧收到的FTS计数  链接培训，  5.0 GT/s链接速度。控制器传输这么多FTS序列，  退出时的序列  当以5.0 GT/s速度运行时，L0 S状态。 | 0x0 |
| 二十三点十六分 | Gen3的接收FTS计数 | RO | 期间从另一侧收到的FTS计数  链接培训，  8.0 GT/s链接速度。控制器传输这么多FTS序列，  退出时的序列  当以8.0 GT/s的速度运行时，L0 S状态。 | 0x0 |
| 三十一点二十四分 | Gen4接收的FTS计数 | RO | 期间从另一侧收到的FTS计数  链接培训，  16.0 GT/s链接速度。控制器传输这么多FTS序列，  退出时的序列  当以16.0 GT/s的速度运行时，L0 S状态。 | 0x0 |

* + - 1. 物理层缓存状态寄存器3 i\_regf\_ip\_pl/i\_dbg\_sts\_3，寄存器偏移：0x 100042 C接收FTS计数状态寄存器。

表14.4：物理层UART状态寄存器3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特位置 | 寄存器字段 | Attr | 描述 | 复位 |

表14.4：物理层调试状态寄存器3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 七比零 | Gen5的接收FTS计数 | RO | 期间从另一侧收到的FTS计数  链接培训，  32.0 GT/s链接速度。控制器传输这么多FTS序列，  退出时的序列  当以32.0 GT/s的速度运行时，L0 S状态。 | 0x0 |

* + 1. 链接错误状态寄存器

如果在链路上发现错误，则可以读取以下寄存器，以检查控制器检测到的错误的性质

* + - 1. 物理层错误状态寄存器4 i\_regf\_ip\_pl/i\_dbg\_sts\_4，寄存器偏移：0x 1000430包括物理层错误状态。

表14.5：物理层UART状态寄存器4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 0 | TLP PHY错误状态 | RO | 此位表示检测  PIPE\_RX\_STATUS  在TLP中填写1以清除此字段。 |  |
| 1 | 跳过操作系统 | RO | 此位表示在接收到命令集块锁定后立即接收到命令集块锁定。  这是一个简单的方法。  成帧错误填写1以清除此字段。 |  |
| 2 | EDS后非法操作系统块 | RO | EDS之后的有效OS块如果有的  其他OS块是  如果在EDS之后收到，则为帧错误，该位置位。 |  |
| 3 | EDS后的数据块 | RO | 如果在EDS之后接收到数据块，则该位置1。写一个1清除  这个错误。 |  |
| 4 | 收到操作系统块，但未使用EDS | RO | 如果接收到有序集块，  没有EDS。这是一个框架错误。写入1以清除此错误。 |  |

表14.5：物理层UART状态寄存器4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 5 | 检测到Gen3成帧错误 | RO | 如果在Gen3中检测到成帧错误，则该位置1。例如，如果无效  如果在数据流中接收到令牌，则会标记此错误。将1写入  清除这个错误。 |  |
| 6 | SDS后收到的操作系统块 | RO | 如果在SDS之后接收到SDS，则该位置1。这是一个框架错误。  写一个1来清除这个呃-  误差 |  |
| 7 | 无效同步标头错误 | RO | 如果检测到无效的同步报头，则该位置1。00和11无效  同步标题。写入1  到  清除这个错误。 |  |
| 8 | 块对齐错误 | RO | 如果PHY在数据流期间丢失块对齐，则该位置1。  这是基于数据流期间意外的PIPE\_RX\_VALID输入解除断言而检测到的。写入1以清除此错误。 |  |
| 9 | 抗扭斜FIFO溢出错误 | RO | 该控制器实现了一个80符号的FI- FO跨通道抗扭斜。在最初完成抗扭斜之后，可能存在由于跨通道的抖动而引起的ppm变化。如果歪斜+ ppm变异，灰超过80个符号的限制，那么抗扭斜FIFO溢出发生。溢出状态在此寄存器中捕获。写1以清除此错误。 |  |

* + - 1. AER i\_uncorr\_err\_status i\_AER\_cap\_struct/i\_uncorr\_err\_status，寄存器偏移：0x 0000104描述：无法纠正的错误状态寄存器。

此寄存器提供PCI Express控制器检测到的各种不可纠正错误的状态软件可以通过将1写入相应的位位置来清除任何错误位。

表14.6：不可纠正的错误状态寄存器

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| BitLocation | 寄存器字段 | Attr | 描述 | 复位 |
| 22 | 无法纠正的内部错误状态。 | 沃什 | 无法纠正的内部错误状态。当控制器检测到内部不可纠正错误（HAL奇偶校验错误）  错误或从任何RAM读取时不可纠正的ECC错误）。此位也在响应客户端签署内部ER时设置，  错误通过输入UNCORRECTABLE\_ER  ROR\_IN.此错误不是特定于函数的。默认情况下，此错误被粘粘的。 |  |
| 20 | 不支持的请求错误状态。 | 沃什 | 不支持的请求错误状态。当控制器收到来自链路  它不支持。此错误不是特定于函数的。默认情况下，此错误被视为非致命错误。在PCI Express规范第6.2.3.2.4.1节中描述的特殊情况下，通过发送ERR\_COR消息报告错误。在所有其他情况下，通过发送ERR\_NONFATAL消息来报告错误。导致错误的已接收请求的标头为  记录在标题日志寄存器中。粘粘的。 |  |
| 19 | 错误状态。 | 沃什 | 错误状态。当控制器检测到  一个ECRC错误，  接收的TLP。此错误不是函数特定的。接收到的有错误的TLP的报头是  记录在标题日志寄存器中。粘粘的。 |  |

表14.6：不可纠正的错误状态寄存器

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| BitLocation | 寄存器字段 | Attr | 描述 | 复位 |
| 18 | 格式错误的TLP状态。 | 沃什 | 格式错误的TLP状态。当控制器接收到  变形的TLP，  链接。此错误不是特定于函数的。默认情况下，此错误被认为是致命的，并通过发送ERR\_FATAL消息来报告.接收到的有错误的TLP的报头是  记录在标题日志寄存器中。粘粘的。 |  |
| 17 | 接收器溢出状态。 | 沃什 | 接收器溢出状态。  当控制器收到违反当前可用接收信用的TLP时，该位置1。此错误不是特定于函数的。粘粘的。 |  |
| 16 | 意外完成状态 | 沃什 | 意外的完成状态。  当控制器从链路收到意外的完成数据包时，该位被设置。此错误不是特定于函数的。粘粘的。 |  |
| 15 | 完成程序中止状态。 | 沃什 | 完成程序中止状态。  当控制器重新打开完成器  中止（CA）状态到从链接接收的请求.此错误是特定于函数的。导致错误的已接收请求的标头记录在标头日志寄存器中。粘粘的。 |  |
| 14 | 完成状态。 | 沃什 | 完成状态。  当与未完成请求相关的完成计时器超时时，该位置1。此错误是函数特定的。默认情况下，此错误被视为非致命错误粘粘的。 |  |

表14.6：不可纠正的错误状态寄存器

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| BitLocation | 寄存器字段 | Attr | 描述 | 复位 |
| 13 | 流控制协议错误状态。 | 沃什 | 流控制协议错误状态。当控制器检测到流量控制协议的某些违规行为时，该位被设置。（i）收到InitFC/UpdateFC DLLP，该DLLP向发送器发出超过2047个累积未用信用用于数据有效载荷或127个用于报头。(ii)InitFC\_P以小于128 B的有效载荷信用被接收，或者（iii）InitFC\_CPL以小于128 B的有效载荷信用被接收。 如果在信用类型的VC上看到FC错误，则在该VC/信用类型的子目录错误将不会被报告，直到通过良好的DLLP、VC停用、链路断开或重置来清除第一个错误。此错误不是特定于功能的粘滞。 |  |
| 12 | 中毒TLP状态 | 沃什 | 中毒TLP状态。当控制器接收到  有毒的TLP  链接.此错误是特定于函数的。默认情况下，此错误被视为非致命错误。通过发送ERR\_NONFATAL消息报告错误。  圣人接收到的有错误的TLP的报头粘粘的。 |  |

表14.6：不可纠正的错误状态寄存器

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| BitLocation | 寄存器字段 | Attr | 描述 | 复位 |
| 4 | 数据链路协议错误状态。 | 沃什 | 数据链路协议错误状态。  当控制器接收到序列号与未确认的TLP或最后确认的TLP的序列号不对应的ACK或Nak DLLP时，该位被设置（有关详细信息，请参阅PCI Ex-  按下基础规范1.1，第3.5.2节）。此错误不是特定于函数的，而是由函数0报告的。粘粘的。 |  |
|  |  |  |  |  |

* + - 1. AER i\_corr\_err\_statusi\_AER\_cap\_struct/i\_corr\_err\_status，寄存器偏移：0x 0000110

描述：此寄存器提供PCI Express控制器检测到的各种可纠正错误的状态软件可以通过将1写入相应的位位置来清除任何错误位。

表14.7：可纠正错误状态寄存器

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 15 | 标头日志寄存器溢出 | R/WOW粘性 | 此位在头日志寄存器溢出时设置，即头无法记录  因为它被前一个标头占用。粘粘的。 |  |
| 14 | 可更正的内部错误状态 | R/WOW粘性 | 当控制器检测到一个内部错误时，该位被设置。  rectable错误条件（一个可纠正的ECC错误，而从任何RAM读取）。该位也在响应客户端通过输入信号  可纠正错误  R\_IN。此错误不是特定于函数的。粘粘的。 |  |

表14.7：可纠正错误状态寄存器

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 13 | 建议性非致命错误状态 | R/WOW粘性 | 当发生不可纠正的错误时，该位被设置，该错误被确定为属于PCI Express 2.0规范第6.2.3.2.4节中描述的特殊情况之一。这会导致控制器生成ERR\_COR消息，而不是ERR\_NONFATAL消息。  圣人粘粘的。 |  |
| 12 | 重播计时器状态 | R/WOW粘性 | 当Da中的重放定时器  控制器的链路层超时，导致控制器重新传输TLP。此错误不是特定于函数的。粘粘的。 |  |
| 8 | 重放屏幕翻转状态 | R/WOW粘性 | 在控制器的数据链路层，当TLP三次重传后重放计数翻转时，该位置1  此错误不是特定于功能的粘滞。 |  |
| 7 | 错误的DLLP状态 | R/WOW粘性 | 当在接收到的DLLP中检测到LCRC错误且物理层未检测到错误时，该位置1。此错误不是特定于功能的。粘粘的。 |  |
| 6 | 不良TLP状态 | R/WOW粘性 | 此位设置时，一个错误是由控制器的数据链路层接收TLP检测。导致此错误的条件是：  (i)LCRC错误（ii）数据包以EDB符号终止，但其LCRC字段不等于计算的CRC的反相值。此错误不是特定于函数的。粘粘的。 |  |

表14.7：可纠正错误状态寄存器

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 0 | 接收器错误状态 | R/WOW粘性 | 当在控制器物理层的接收端检测到错误时，此位被设置（例如，位错误或编码违规）。此位设置后，任何以下错误：（1）物理层报告8B 10 B错误，双奇偶校验错误，弹性缓冲区溢出错误，下流错误（2）GEN 3 TLP，DLLP成帧错误（3）OS块接收与出EDS（4）数据块接收后EDS（5）非法OS块后EDS（6）OS块接收后跳过OS（7）OS块接收后，ter SDS（8）同步头- er错误（9）损失的Gen 3块对齐此错误是不功能specif- ic。粘粘的。 |  |

* + - 1. i\_ltssm\_recovery\_count\_reg\_[n]i\_regf\_ip\_pl/i\_ltssm\_recovery\_count\_reg\_0，寄存器偏移：0x 10006 F4

描述：此寄存器包含LTSSM以2.5GT/s速度从L0转换到Recovery的次数计数

表14.8：LTSSM恢复计数寄存器

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 比特 | 接入 | 复位值 | 描述 |
| [RMTCNT]  远程启动的L0再加工转换计数。 | 三十一点十六分 | 沃什 | 0 | 远程启动的L0再加工转换计数。次数计数  LTSSM过渡到  由于远程端而进行恢复。  写所有的1可以清除这个。 |
| [LCLCNT]  本地启动的L0检索转换计数。 | 十五比零 | 沃什 | 0 | 本地启动的L0检索转换计数。  由于本地RX错误，LTSSM从L0本地转换到Recovery的次数计数写所有的1可以清除这个。 |

Gen 2、Gen 3、Gen 4、Gen 5和Gen 6实现了类似的L0->恢复转换计数寄存器

**Gen 2：i\_regf\_ip\_pl/i\_ltssm\_recovery\_count\_reg\_1，**寄存器偏移：0x 10006 F8

**Gen 3：i\_regf\_ip\_pl/i\_ltssm\_recovery\_count\_reg\_2，**寄存器偏移：0x 10006 FC

**Gen 4：i\_regf\_ip\_pl/i\_ltssm\_recovery\_count\_reg\_3，**寄存器偏移：0x 1000700**Gen 5：i\_regf\_ip\_pl/i\_ltssm\_recovery\_count\_reg\_4，**寄存器偏移：0x 1000704**Gen 6：i\_regf\_ip\_pl/i\_ltssm\_recovery\_count\_reg\_5，**寄存器偏移：0x 1000708

* + - 1. i\_ltssm\_recovery\_count\_local\_countries\_regi\_regf\_ip\_pl/i\_ltssm\_recovery\_count\_local\_countries\_reg，寄存器偏移：0x 10006 F0

描述：此寄存器控制i\_ltssm\_recovery\_count\_reg中本地启动的LTSSM从L0到恢复转换下将计数的事件

表14.9：LTSSM恢复计数本地控制寄存器

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 比特 | 接入 | 复位值 | 描述 |
|  |  |  |  | 本地LTSSM恢复位图- |
|  |  |  |  | 控制将在i\_ltssm\_recovery\_count\_reg中计数的事件，用于本地启动的从L0到Recovery的LTSSM转换。 |
|  |  |  |  | 编码是： |
|  |  |  |  | 1：启用事件计数  i\_ltssm\_recovery\_count\_r  eg. |
|  |  |  |  | 0：未启用事件计数  i\_ltssm\_recovery\_count\_r例如 |
|  |  |  |  | 位[0]：RP从CFG重训练链路 |
|  |  |  |  | 位[1]：EP启动的速度改变再训练链接 |
|  |  |  |  | 位[2]：EP 8GT请求EQ再训练链接 |
|  |  |  |  | 位[3]：EP 16 GT请求EQ重新训练链路位[4]：EP 32 GT请求EQ重新训练链路位[5]：RP发起的禁用链路请求 |
|  |  |  |  | 位[6]：RP发起的热复位请求 |
| [LCLTSRCVBM]  本地Ltssm恢复位图 | 三十一比零 | RW | 00 | 位[7]：RP发起的环回请求  bit[8]：NFTS位[9]：RP自动均衡速度更改为  Gen3及更高版本 |
|  |  |  |  | 位[10]：RP自动速度更改为Gen 1位[11]：上层启动L1条目 |
|  |  |  |  | 位[12]：上层启动的L2条目 |
|  |  |  |  | 位[13]：错误情况：在没有EIOS的情况下 |
|  |  |  |  | 位[14]：错误情况：PIPE\_RX\_VALID在L0 |
|  |  |  |  | 位[15]：保留。 |
|  |  |  |  | bit[16]：DL Replay Rollover Retrain Link Request |
|  |  |  |  | 位[17]：成帧错误重训练链接 |
|  |  |  |  | bit[18]：FC重训练链接 |
|  |  |  |  | 位[19]：EP/RP链路启动-配置重新训练链路 |
| adence机密 |  |  |  | 位[20]：保留 142  位[21]：LM发起的IORECAL请求重新- |

* + 1. 抗扭斜状态寄存器
       1. 物理层缓存状态寄存器5

以下寄存器记录控制器在不同速度下检测到的偏斜：i\_regf\_ip\_pl/i\_dbg\_sts\_5，寄存器偏移：0x 1000434

包括物理层去偏斜状态。

表14.10：物理层UART状态寄存器5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 六比零 | Gen1偏斜 | RO | 此字段指示在Gen1的通道  反映Gen 1上最近进入配置.空闲/恢复.空闲期间的偏斜。 | 0 |
| 7 | Gen1偏斜有效 | RO | 此位验证Gen1偏斜。  在Gen1完成抗扭斜时设置为1 | 0 |
| 十四点八分 | Gen2偏斜 | RO | 此字段指示在Gen2的通道  反映Gen 2上最近进入配置.空闲/恢复.空闲期间的偏斜。 | 0 |
| 15 | Gen2偏斜有效 | RO | 此位验证Gen2偏斜。  在Gen2完成抗扭斜时，设置为1 | 0 |
| 二十二点十六分 | Gen3偏斜 | RO | 此字段指示在Gen3通道  反映Gen 3上最近进入配置.空闲/恢复.空闲期间的偏斜。 | 0 |
| 23 | Gen3偏斜有效 | RO | 此位验证Gen3偏斜。  在Gen3完成抗扭斜时，设置为1 | 0 |
| 三十点二十四分 | 第四代倾斜 | RO | 此字段指示在Gen 4通道  反映Gen 4上最近进入配置.空闲/恢复.空闲期间的偏斜。 | 0 |
| 31 | Gen4偏斜有效 | RO | 此位验证Gen4偏斜。  在Gen4完成抗扭斜时，设置为1 | 0 |

* + - 1. 物理层去偏斜状态寄存器6 i\_regf\_ip\_pl/i\_dbg\_sts\_6，寄存器偏移：0x 1000438包括物理层去偏斜状态。

表14.11：物理层UART状态寄存器6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 六比零 | Gen5偏斜 | RO | 此字段指示在Gen 5的通道  反映Gen 5上最近进入配置.空闲/恢复.空闲期间的偏斜。 | 0 |
| 7 | Gen5偏斜有效 | RO | 此位验证Gen5偏斜。  在Gen5完成抗扭斜时设置为1 | 0 |
| 十四点八分 | Gen6偏斜 | RO | 此字段指示在Gen 6通道  反映最近进入Gen 6的Config.Idle/Recovery.Idle | 0 |
| 15 | Gen6偏斜有效 | RO | 此位验证Gen6偏斜。  在Gen6完成抗扭斜时设置为1 | 0 |

* + 1. 链路均衡控制设置寄存器

控制器用于执行均衡的算法可使用以下寄存器进行编程

* + - 1. Gen 3均衡控制寄存器0i\_regf\_ip\_pl/i\_g3\_eq\_ctrl\_0，寄存器偏移：0x 1000474

表14.12：Gen3均衡控制寄存器0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 位  位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 十比零 | 支持的Gen3预设 | RW | 此寄存器可用于对lo支持的预设进行编程，  校准发射机在8Gbps。  此注册器的默认值由  预设置  带输入。补注：应  8.0 GT/s及更高，必须支持全回转  信号减小摇摆  信令必须实现信令#4，#1，#9，#5，  #6和#3。 |  |
| 十四点十一分 | 默认Gen3发送器预设 | RW | 使用默认Gen3变送器预设值  对于在Recovery.RcvrCfg中未接收到EQ TS2  LTSSM状态 |  |
| 十七点十五分 | 默认Gen3接收器预设提示 | RW | 默认Gen3接收器预设提示值用于未重新  接收EQ TS2，  恢复.RcvrCfgLTSSM状态 |  |
| 二十点十八分 | Gen3最大评估收敛计数 | RW | 控制具有方向  00s的更改反馈  在推断均衡收敛之前。0：推断1  反馈000000 1：  000000 2的2次反馈后推断收敛：3次反馈  连续反馈  的000000 .. 7：在8次连续反馈后推断收敛  000000.注：每条泳道  独立计数连续反馈000000。注：计数为  非000000后重置  每条车道上的反馈。 |  |

表14.12：Gen3均衡控制寄存器0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 位  位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 21 | EP 8GT请求均衡再训练链路 | RW | 端点设备固件可使用此位请求8 GT/s Equal-  化重做。此位可  可以在链路启动后随时设置。将1写入此字段将导致  控制器要过渡-  tion  恢复。TS2中的请求均衡位  有序集将在Recovery.Rcvr.Cfg中设置为1，以请求均衡  八辆GT 这一点是自动-  清除  通过控制器的内部逻辑，  重新训练已经完成并且链路已经到达L0状态。  这一点也是自...  清除  如果不是在Gen3或Gen4中。设备固件必须  在任何后续的重训练请求之前等待该位被 |  |
| 22 | 静止保证8GT | RW | 此位可用于对TS2  在Recovery.Rcvr.Cfg  在8 GT/s请求均衡期间状态。 |  |

表14.12：Gen3均衡控制寄存器0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 位  位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 二十六点二十三分 | 最大8GT均衡请求限制 | RW | 8 GT/s均衡请求的数量必须  根据PCIe规范进行限制。这个寄存器  可用于编程的最大数量  由端点自动发起的8 GT/s  0000：自动8 GT/s  均衡请求已禁用。0001：自动化  8 GT/s均衡请求限制为1。0010：自动8 GT/s相等  化请求限制为  第1111章：一个人的世界  8 GT/s均衡请求  极限是15， | 0 |
| 27 | 禁用最大值迭代 | RW | MAX\_EVAL\_ITERATION  输入控制均衡  阶段2（EP）或阶段3（RP）。  此位可用于屏蔽  最大值迭代  并执行更多的迭代，直到  收敛或24 ms超时。 | 0 |

* + - 1. Gen 3均衡控制寄存器1i\_regf\_ip\_pl/i\_g3\_eq\_ctrl\_1，寄存器偏移：0x 1000478

表14.13：Gen3均衡控制寄存器1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 位  位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 三比零 | Gen3均衡通道选择 | RW | 选择通道以检查Gen3通道均衡状态寄存器0和Gen3通道均衡状态寄存器1中的状态。 | 0 |

表14.13：Gen3均衡控制寄存器1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 位  位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 4 | Gen3 EQ Phase2远程发射预设使能 | RW | 仅用于EP模式。该位使控制器能够反馈  Tx预设，用于重新-  在链路均衡阶段2的第一次迭代中，以Gen3的速度发送微尘端  用于反馈的Gen3 Tx预设可通过该寄存器的位[4：1]进行编程。  为RP模式保留。 | 0 |
| 5 | Gen3本地以太网Tx预设启用 | RW | 这是一个调试位。可用于EP和RP模式。  如果启用，则控制器在整个Gen3期间将Gen3本地收发器预设应用于本地发射器。  控制器执行系数映射预置，然后继续驱动  PIPE\_TX\_DEEMPHASIS  PIPE接口信号。 | 0 |
|  |  |  |  |  |
| 十一点八分 | Gen3 EQ Phase2远程发射预设 | RW | 仅用于EP模式。当使用位0启用时，此Tx预设将  在TS1  用于第三代产品第二阶段的第一个版本中的远端变送器。  为RP模式保留。 | 0 |
|  |  |  |  |  |
| 下午15点12分 | 第3代本地射频Tx预设 | RW | 这是一个调试寄存器字段。可用于EP和RP模式。  当使用位5启用时，此Tx预设将始终应用于本地发射机  Gen3不考虑Gen3均衡。 | 0 |

* + - 1. 第4代均衡控制寄存器0i\_regf\_ip\_pl/i\_g4\_eq\_ctrl\_0，寄存器偏移：0x 1000488

表14.14：Gen4均衡控制寄存器0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 位  位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 十比零 | 支持的Gen4预设 | RW | 此寄存器可用于对lo支持的预设进行编程，  校准发射机16 GT/  S.  此注册器的默认值由  预设置  带输入。补注：应  8.0 GT/s及更高，必须支持全回转  信号减小摇摆  信令必须实现信令#4，#1，#9，#5，  #6和#3。 |  |
| 十四点十一分 | 默认Gen4发送器预设 | RW | 使用默认Gen4变送器预设值  对于在恢复中未接收到8G EQ TS2的通道。  LTSSM状态 |  |
| 二十点十八分 | Gen4最大评估收敛计数 | RW | 控制具有方向  00s的更改反馈  在推断均衡收敛之前。0：推断1  反馈000000 1：  000000 2的2次反馈后推断收敛：3次反馈  连续反馈  的000000 .. 7：在8次连续反馈后推断收敛  000000.注：每条泳道  独立计数连续反馈000000。注：计数为  非000000后重置  每条车道上的反馈。 |  |

表14.14：Gen4均衡控制寄存器0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 位  位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 21 | EP 16 GT请求均衡再训练链路 | RW | 端点设备固件可使用此位请求16 GT/s  均衡重做。这  位可以  可以在链路启动后随时设置。将1写入此字段将导致  主计长转  tion  恢复。TS2中的请求均衡位  有序集将在Recovery.Rcvr.Cfg中设置为1，以请求均衡  16辆GT这一点是自动-  清除  通过控制器的内部逻辑，  重新训练已经完成并且链路已经到达L0状态。  这一点也是自...  清除  如果不是在Gen3或Gen4中。设备固件必须  在任何后续的重训练请求之前等待该位被 |  |
| 22 | 静止保证16 GT | RW | 此位可用于对TS2  在Recovery.Rcvr.Cfg  在16 GT/s请求均衡期间状态。 |  |

表14.14：Gen4均衡控制寄存器0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 位  位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 二十六点二十三分 | 最大16 GT均衡请求限制 | RW | 16 GT/s均衡请求的数量  根据PCIe规范进行限制。这个寄存器  可用于编程16 GT/s均衡  自动请求  由端点发起。0000：自动16 GT/s均衡请求  禁用.第0001章：一个人  IC  16 GT/s均衡请求限制为1。0010：自动16 GT/s相等-  化请求限制为  第1111章：一个人的世界  16 GT/s均衡要求  极限是15， |  |

* + - 1. 第4代均衡控制寄存器1i\_regf\_ip\_pl/i\_g4\_eq\_ctrl\_1，寄存器偏移：0x 100048 C

表14.15：Gen4均衡控制寄存器1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 位  位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 三比零 | Gen4均衡通道选择 | RW | 选择通道以检查Gen4通道均衡状态寄存器0和Gen4通道均衡状态寄存器1中的状态。 | 0 |
|  |  |  |  |  |
| 4 | Gen4 EQ Phase2远程发射预设使能 | RW | 仅用于EP模式。该位使控制器能够反馈  Tx预设，用于重新-  在Gen4速度下链路均衡阶段2的第一次迭代中的微尘端发射机  用于反馈的Gen4 Tx预设可通过该寄存器的位[15：12]进行编程。 | 0 |

表14.15：Gen4均衡控制寄存器1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 位  位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 5 | Gen4 EP 8GT EQ TS2  使 | RW | 仅用于EP模式。在Gen3至Gen4变速谈判期间，  此位使能  EP模式下的控制器在恢复.接收.配置状态下传输8G EQ TS2，而不是PCIe规范中定义的标准TS2。  将用于8GT EQ TS2的Tx预设可通过该寄存器的位[15：12]进行编程。 | 0 |
| 6 | 第4代本地Tx预设启用 | RW | 这是一个调试位。可用于EP和RP模式。  如果启用，则控制器在整个Gen4期间将Gen4本地收发器预设本地应用于本地发射器。  控制器执行系数映射预置，然后继续驱动  PIPE\_TX\_DEEMPHASIS  PIPE接口信号。 | 0 |
| 十一点八分 | Gen4 EQ Phase2远程发射预设 | RW | 仅用于EP模式。  如果启用，此Tx预设将在第3代至第4代速度变化NE-EXTRATION期间在8 G EQ TS 2s中传输。  此外，如果启用，此Tx预设将在第四代均衡阶段2的第一次迭代中以TS1s发送。  注：如果远端设备在均衡阶段2开始时通告相同的预设，则将跳过此预设。 | 0 |
| 15点12分 | 第4代本地射频Tx预设 | RW | 这是一个调试寄存器字段。可用于EP和RP模式。  启用时，使用  位-16，该Tx预设将在整个Gen 4期间应用于本地发射机，而不管Gen 4均衡。 | 0 |

* + - 1. 第5代均衡控制寄存器0i\_regf\_ip\_pl/i\_g5\_eq\_ctrl\_0，寄存器偏移：0x 100049 C

表14.16：Gen5均衡控制寄存器0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 位  位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 十比零 | 支持的Gen5预设 | RW | 此寄存器可用于对lo支持的预设进行编程，  32 GT时的校准变送器/  S.  此注册器的默认值由  预设置  带输入。补注：应  8.0 GT/s及更高，必须支持全回转  信号减小摇摆  信令必须实现信令#4，#1，#9，#5，  #6和#3。 |  |
| 十四点十一分 | 默认Gen5发送器预设 | RW | 使用默认Gen5变送器预设值  对于在恢复中未接收到8G EQ TS2的通道。  LTSSM状态 |  |
| 十六点十五分 | Gen5最大评估收敛计数 | RW | 控制具有方向  00s的更改反馈  在推断均衡收敛之前。0：推断1  反馈000000 1：  000000 2的2次反馈后推断收敛：3次反馈  连续反馈  的000000 .. 7：在8次连续反馈后推断收敛  000000.注：每条泳道  独立计数连续反馈000000。注：计数为  非000000后重置  每条车道上的反馈。 |  |

表14.16：Gen5均衡控制寄存器0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 位  位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 17 | EP 32 GT请求均衡再训练链路 | RW | 端点设备固件可使用此位请求32 GT/s  均衡重做。这  位可以  可以在链路启动后随时设置。将1写入此字段将导致  主计长转  tion  恢复。TS2中的请求均衡位  有序集将在Recovery.Rcvr.Cfg中设置为1，以请求均衡  32辆GT这一点是自动-  清除  通过控制器的内部逻辑，  重新训练已经完成并且链路已经到达L0状态。  这一点也是自...  清除  如果不是在第三代或第四代或第五代。设备固件必须  在任何后续的重训练请求之前等待该位被 |  |
| 18 | 静止保证32GT | RW | 此位可用于对TS2  在Recovery.Rcvr.Cfg  在32 GT/s请求均衡期间状态。 |  |

表14.16：Gen5均衡控制寄存器0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 位  位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 二十二点十九分 | 最大32 GT均衡请求限制 | RW | 32 GT/s均衡请求的数量  根据PCIe规范进行限制。这个寄存器  可用于编程最大均衡  自动请求  由端点发起。0000：自动32 GT/s均衡请求  禁用.第0001章：一个人  IC  32 GT/s均衡请求限制为1。0010：自动32 GT/s相等-  化请求限制为  第1111章：一个人的世界  32 GT/s均衡要求  极限是15， |  |

* + - 1. Gen 5均衡控制寄存器1i\_regf\_ip\_pl/i\_g5\_eq\_ctrl\_1，寄存器偏移：0x 10004 A0

表14.17：Gen5均衡控制寄存器1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 位  位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 三比零 | Gen5均衡通道选择 | RW | 选择通道以检查Gen5通道均衡状态寄存器0和Gen5通道均衡状态寄存器1中的状态。 | 0 |
| 4 | Gen5 EQ Phase2远程发射预设使能 | RW | 仅用于EP模式。该位使控制器能够反馈  Tx预设，用于重新-  在链路均衡阶段2的第一次迭代中，以Gen5的速度发送微尘端  用于反馈的Gen5 Tx预设可通过该寄存器的位[26：23]进行编程。 | 0 |

表14.17：Gen5均衡控制寄存器1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 位  位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 5 | Gen5 EP EQ TS2启用 | RW | 仅用于EP模式。在Gen4至Gen5变速谈判期间，  此位使能  控制器在EP模式下传输EQ TS 2（或128 b/130 b EQ TS 2），  Recovery. Rcvr. Cfg状态，而不是PCIe规范中定义的标准TS2。  EQ TS2中使用的Tx预设可通过该寄存器的位[26：23]进行编程。 | 0 |
| 6 | Gen5本地Tx预设启用 | RW | 这是一个调试位。可用于EP和RP模式。  如果启用，则控制器在整个Gen5期间将Gen5本地收发器预设应用于本地发射器。  控制器执行系数映射预置，然后继续驱动  PIPE\_TX\_DEEMPHASIS  PIPE接口信号。 | 0 |
| 十一点八分 | Gen5 EQ Phase2远程发射预设 | RW | 仅用于EP模式。如果启用，此Tx预设将在  EQ TS 2（或128 b/130 b  EQ TS2）在与Gen5的速度变化协商期间。  此外，如果启用，此Tx预设将在第5代均衡阶段2的第一次迭代中以TS1s发送。  注：如果远端设备在均衡阶段2开始时通告相同的预设，则将跳过此预设。 | 0 |

表14.17：Gen5均衡控制寄存器1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 位  位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 下午15点12分 | Gen5本地发射机发射预设 | RW | 这是一个调试寄存器字段。可用于EP和RP模式。  启用时，使用  位-16，该Tx预设将在整个Gen 5期间应用于本地发射机，而不管Gen 5均衡。 | 0 |

* + - 1. 均衡控制寄存器0i\_regf\_ip\_pl/i\_eq\_ctrl\_0，寄存器偏移：0x 1000470

表14.18：均衡控制寄存器0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 位  位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 0 | RXEQINPROGRASS  中止禁用 | RW | 在PIPE\_PCLK因均衡错误而停止的意外情况下，此位可设置为  将RxEqIn- Progress与其余的  均衡状态机。这一点不应该  设置为正常使用。 | 0 |

表14.18：均衡控制寄存器0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 位  位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 二比一 | RXEQINPROGRASS  中止计时器模式控制 | RW | 当LTSSM均衡阶段2中发生24 ms超时时，  控制器中止  均衡阶段2并过渡到恢复.接收.锁定。在  在这种情况下，RxEqEval  PIPE接口上的输出将立即被取消断言（如果它是-  serted）。的  RxEqInProgress输出将保持高电平并等待PhyStatus脉冲。康-  控制器实现定时器  选择上限，以便在中止  取消断言RxEqIn-  中求进工作总  00：等待最多4个PIPE\_PCLK周期。01：等待最大值  8 PIPE\_PCLK周期。  10：等待最多16个PIPE\_PCLK  期11：禁用。等待接收到PhyStatus脉冲。注意：仅在以下情况下  当在Equal中发生LTSSM 24 ms超时时，RxEqEval被断言-  化。 | 0 |
| 3 | 禁用LOCALLF、LOCALFS  变速后取样 | RW | 根据PIPE 4.2规范，LOCALLF、LOCALFS输出来自  PHY可采样  复位#后的uponf PHYSTATUS脉冲或第一个 PHYSTATUS脉冲  变速后脉冲  到  GEN3.此位可设置为1，以在速度更改为  GEN3或更高版本 | 0 |

表14.18：均衡控制寄存器0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 位  位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 4 | 反馈错误后启用 | RW | 当控制器从PHY接收到方向改变反馈时，反馈是  应用于当前系数并检查错误。  在验证阶段2（EP）或  阶段3（RP），1次迭代被定义为设备通过PCIe链路在TS 1中向远端发出的1组反馈。  此位控制方向改变错误时的迭代装置反馈：  0：如果反馈无效，则控制器断言  管道\_无效请求  T到PHY，丢弃反馈并且不在同一迭代中重试PIPE\_RXEQEVAL。  在迭代过程中系数设置保持重试  下一次迭代中的PIPE\_RXEQEVAL  1：如果反馈无效，则控制器断言  管道\_无效请求  T到物理层，放弃反馈并在中  相同迭代  直到接收到有效的反馈。 | 0 |

表14.18：均衡控制寄存器0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 位  位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 5 | 禁用LOCALLF，LO-CALFS采样后，速率变化的环回. |  | 按照管道规格，灰，LOCALLF，LO- CALFS从物理层输出可以进行采样后，复位#或在第一PHYSTATUS脉冲PHYSTATUS脉冲后，速率变化到GEN 3。速率改变  当当前链路速率与环回速率不同时，在环回进入期间启动。此位可设置为1，以便在环回期间 |  |
| 6 | 禁用杂散管道本地TX系数有效检查。 |  | 根据PIPE规范，pipe\_localtxcoefficientsvalid输入预计将  仅在MAC查询预设为pipe\_getlocalpresetcoeffients上的系数映射之后才被断言。控制器检查并丢弃pipe\_localtxcoefficientsvalid输入，如果没有  在pipe\_getlocalpresetcoeffi cients上发出请求。0：丢弃pipe\_localtxcoefficientsvalid输入上的虚假响应。1：商店  pipe\_localTxcoefficientsva上的所有响应都将被输入并用于任何未来的预设查询。 |  |
| 三十一比七 | 保留 |  |  |  |

* + - 1. IORECAL控制状态寄存器i\_regf\_ip\_pl/i\_iorecal\_cnc\_stat，寄存器偏移：0x 10006 B 0

控制器的PIPE IO重新校准功能可使用此寄存器进行控制PIPE IORECAL信号的状态也反映在该寄存器中。

表14.19：IORECAL控制状态寄存器

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 位  位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 五比零 | 控制器启动IORE- CAL禁用 | RW | 此字段用于禁用控制器启动的IORECAL后，接收器/帧错误在每个速度。  （1：禁用，0：启用）。位[0] -保留  位[1] -保留  位[2] -在GEN 3禁用IOREA- CAL。  位[3] -在GEN 4禁用IOREA- CAL。  位[4] -在GEN 5禁用IOREA- CAL。  位[5] -在GEN 6禁用IOREA- CAL。 | 0 |
| 七点六分 | 保留 |  |  |  |
| 十三点八分 | PHY发起的IORECAL掩码 | RW | 该字段用于屏蔽每个速度下PHY发起的IORECAL请求。  （1）掩码PHY IORECAL  请求，  0：处理PHY IORE-CAL请求）。  位[0] -屏蔽GEN 1的pipe\_iorecal\_request。  位[1] -在GEN 2屏蔽pipe\_iorecal\_request。  位[2] -在GEN 3处屏蔽pipe\_iorecal\_request。  位[3] -在GEN 4屏蔽pipe\_iorecal\_request。  位[4] -在GEN 5屏蔽pipe\_iorecal\_request。  位[5] -在GEN 6屏蔽pipe\_iorecal\_request。 | 0 |
| 十五点十三分 | 保留 |  |  |  |

表14.19：IORECAL控制状态寄存器

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 位  位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 16 | IORECAL可重复使用 | RW | 此字段用于启用索引-  等待Recovery.Rcvr.Lock时等待24 ms超时，用于来自PHY的pipe\_iorecal\_done响应。  如果pipe\_iorecal\_done上没有响应，则：1：LTSSM转换  Recovery.Rcvr.Lock->  24 ms超时后恢复. Rcvr. Cfg  0：LTSSM转换恢复接收器锁定-  > Recovery.Rcvr.Cfg/ Recovery.Speed/Configuration/Detect after 24ms timeout as per PCIe specification for 24 ms timeout.  如果pipe\_iorecal\_done花费的时间超过24 ms，则将此位设置为1可防止链路断开。 | 0 |
| 17 | IORECAL临床试验状态 | 沃什 | 此状态位指示在等待pipe\_iorecal\_done响应时是否发生24 ms超时。  注：仅当IORECAL触发器使能（位-16）编程为1时，该位才会置1。 | 0 |
| 18 | IORecal再培训链接 | RW | 将1写入此字段将导致控制器启动L0-> Recovery  转换为IORecal断言。  该位由控制器  IORecal完成，LTSSM转换出Recovery.Rcvr.Lock状态。  软件必须等待该位被清除，然后才能再次设置。 | 0 |
| 二十八点十九分 | 保留 |  |  |  |
| 29 | 管道IORECAL完成  地位 | RO | 此状态位指示pipe\_iorecal\_done输入的当前状态。 | 0 |

表14.19：IORECAL控制状态寄存器

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 位  位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 30 | 管道IORECAL RE-  QUEST状态 | RO | 此状态位指示pipe\_oriecal\_request\_status输入的当前状态。 | 0 |
| 31 | PIPE IORECAL状态 | RO | 此状态位指示pipe\_oriecal输出的当前状态。 | 0 |

* + 1. 链路均衡寄存器

链路均衡阶段的结果反映在状态寄存器中。这些寄存器可用于调试链路均衡过程。

* + - 1. Gen 3通道均衡状态寄存器0i\_regf\_ip\_pl/i\_g3\_ln\_eq\_sts\_0，寄存器偏移：0x 100047 C

表14.20：Gen3通道均衡状态寄存器0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 三比零 | Gen3本地TX预设 | RO | TX预设同意此车道。 |  |
| 4 | Gen3本地TX预设有效 | RO | TX预设有效指示符。此位设置时，  在终端模式下设置use\_preset位时，在TS 1中接收TX预设  第3阶段或RC模式  第二阶段。 |  |
| 七点五 | 保留 |  |  |  |
| 25：8 | Gen3本地TX系数 | RO | 此通道的TX系数已达成一致。[25：20]：发布光标Co-  [19：14]第19话：你是谁？  系数[13：8]：预误差系数 |  |

* + - 1. Gen 3通道均衡状态寄存器1i\_regf\_ip\_pl/i\_g3\_ln\_eq\_sts\_1，寄存器偏移：0x 1000480

表14.21：Gen3通道均衡状态寄存器1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 三比零 | Gen3远程TX预设 | RO | TX预设反馈，如签名到远程TX，用于该通道。 |  |

表14.21：Gen3通道均衡状态寄存器1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 4 | Gen 3远程TX预设有效 | RO | TX预设有效指示符。此位设置时，  TX预置反馈在TS 1中提供，use\_preset位在端点模式阶段2或RC模式阶段3中设置。 |  |
| 22：8 | Gen3远程TX系数 | RO | TX分配给此通道的远程TX的系数反馈。  [25：20]：后光标系数[19：14]：光标系数[13：8]：前光标系数 |  |

* + - 1. 第4代通道均衡状态寄存器0i\_regf\_ip\_pl/i\_g4\_ln\_eq\_sts\_0，寄存器偏移：0x 1000490

表14.22：Gen4通道均衡状态寄存器0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 三比零 | Gen4本地TX预设 | RO | TX预设同意此车道。 |  |
| 4 | Gen4本地TX预设有效 | RO | TX预设有效指示符。此位设置时，  在终端模式下设置use\_preset位时，在TS 1中接收TX预设  第3阶段或RC模式  第二阶段。 |  |
| 七点五 | 保留 |  |  |  |
| 25：8 | Gen4本地TX系数 | RO | 此通道的TX系数已达成一致。[25：20]：发布光标Co-  [19：14]第19话：你是谁？  系数[13：8]：预误差系数 |  |

* + - 1. Gen 4通道均衡状态寄存器1i\_regf\_ip\_pl/i\_g4\_ln\_eq\_sts\_1，寄存器偏移：0x 1000494

表14.23：Gen4通道均衡状态寄存器1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 三比零 | Gen4远程TX预设 | RO | TX预设反馈，如签名到远程TX，用于该通道。 |  |

表14.23：Gen4通道均衡状态寄存器1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 4 | Gen4远程TX预设有效 | RO | TX预设有效指示符。此位设置时，  TX预设反馈在TS 1中提供，并在端点模式阶段2或RC模式阶段3中设置use\_preset位。 |  |
| 七点五 | 保留 |  |  |  |
| 25：8 | Gen4 Remote TX Coeffi- cient | RO | TX分配给此通道的远程TX的系数反馈。  [25：20]：后光标系数[19：14]：光标系数[13：8]：前光标系数 |  |

* + - 1. 第5代通道均衡状态寄存器0i\_regf\_ip\_pl/i\_g5\_ln\_eq\_sts\_0，寄存器偏移：0x 10004 A4

表14.24：Gen5通道均衡状态寄存器0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 三比零 | Gen5本地TX预设 | RO | TX预设同意此车道。 |  |
| 4 | Gen5本地TX预设有效 | RO | TX预设有效指示符。此位设置时，  在终端模式下设置use\_preset位时，在TS 1中接收TX预设  相位3或RC模式  第二阶段。 |  |
| 七点五 | 保留 |  |  |  |
| 22：5 | Gen5本地TX系数 | RO | 此通道的TX系数已达成一致。[25：20]：发布光标Co-  [19：14]第19话：你是谁？  系数[13：8]：预误差系数 |  |

* + - 1. Gen 5通道均衡状态寄存器1i\_regf\_ip\_pl/i\_g5\_ln\_eq\_sts\_1，寄存器偏移：0x 10004 A8

表14.25：Gen5通道均衡状态寄存器1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 三比零 | Gen5远程TX预设 | RO | TX预设反馈，如签名到远程TX，用于该通道。 |  |

表14.25：Gen5通道均衡状态寄存器1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 4 | Gen5远程TX预设有效 | RO | TX预设有效指示符。此位设置时，  TX预置反馈在TS 1中提供，use\_preset位在端点模式阶段2或RC模式阶段3中设置。 |  |
| 七点五 | 保留 |  |  |  |
| 25：8 | Gen5 Remote TX Coeffi- cient | RO | TX分配给此通道的远程TX的系数反馈。  [25：20]：后光标系数[19：14]：光标系数[13：8]：前光标系数 |  |

* + - 1. 均衡FIFO控制寄存器i\_regf\_ip\_pl/i\_eq\_dbg\_fifo\_cycle，寄存器偏移：0x 10004 F0

表14.26：均衡FIFO控制寄存器

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 0 | 清除所有捕获 | RW | 设置此位可清除EQ报警状态  登记册。  如果未设置，则  在状态寄存器中允许捕获。 |  |
| 四比一 | 捕获通道选择 | RW | 将其均衡信息将被封顶的通道进行分组，  tured。  请注意，这表示物理车道号。 |  |
| 六点五 | 捕获EQ相位选择 | RW | 当捕获完成时，停止均衡阶段  01：第2阶段，   1. ：第三阶段， 2. ：第二阶段和第三 |  |
| 九点七分 | 捕获均衡器速度选择 | RW | 捕获均衡器速度选择。设置要执行捕获的链路速度：  000：任何速度，001：Gen3，  010：Gen4，   1. ：Gen5， 2. ：Gen6 |  |

表14.26：均衡FIFO控制寄存器

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 10 | Eq捕获行为 | RW | Eq Capture Be-or.  如果设置了此选项，则捕获  否则捕获最后64个事件 |  |

* + - 1. 均衡FIFO状态0寄存器i\_regf\_ip\_pl/i\_eq\_dbg\_fifo\_sts\_0，寄存器偏移：0x 10004 F4**描述：**

所有的动态均衡信息都记录在这个寄存器中。

这是使用同步FIFO实现的，该FIFO将所有捕获的事件存储为单独的32位条目。每次读取都会增加读指针，客户端必须存储读取的数据FIFO可以通过EQ UART监控器控制寄存器中的清除所有捕获位来清除

表14.27：均衡FIFO状态0寄存器

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 五比零 | 本地FS | RO | 所选通道和速度的本地PHY的FS值 |  |
| 十一点六分 | 本地LF | RO | 选择本地物理层的车道LF值 |  |
| 十七点十二分 | 远程FS | RO | 所选通道和速度的远程PHY的FS |  |
| 二十三点十八分 | 远程LF | RO | 所选通道和速度的远程PHY的LF |  |

* + - 1. 均衡FIFO状态1寄存器i\_regf\_ip\_pl/i\_eq\_dbg\_fifo\_sts\_1，寄存器偏移：0x 10004 F8

表14.28：均衡FIFO状态1寄存器

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 十七比零 | 系数 | RO | 相位0：存储从初始预处理映射的初始本地TX系数。  集第1阶段：库存初始化-  tial  远程系数在第1阶段进行广告。(Cp、LF、FS）、EP Ph2/RC  阶段3：存储电流  远程设备的系数。EP Ph 3/RC Phase 2：存储系数  收到的答复  mote设备。 |  |
| 二十一点十八分 | 预设 | RO | 阶段0：存储接收到的初始本地TX预设  在第0阶段。阶段1：存储初始远程预设  在第一阶段公布。EP Ph 2/RC Phase 3：存储Re  mote设备。EP Ph3/  RC阶段2：存储从远程接收的预设  设备. |  |
| 22 | 预设有效 | RO | 1：预设有效，表示[21：18]有效。Phase0：设置为“1”以指示  初始本地预置  有效。Phase1：设置为“1”以指示所通告的远程预处理  set有效。EP Ph2/RC  Ph3：如果控制器提供预设反馈，则设置为1，系数设置  反馈EP Ph3/RC  Ph2：  反映从远程  端 |  |

表14.28：均衡FIFO状态1寄存器

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 23 | 系数 | RO | 阶段0：如果在阶段0中接收到不支持的预设，则设置为“1”。  阶段1：设置为“0”，因为  没有  在阶段1中拒绝。EP Ph 2/ RC Ph 3：指示远程端断开-  风化组此位表示  的  当前的系数或预设被远程终端设备拒绝。  EP Ph 3/RC Ph 2：独立  凯茨  控制器拒绝了接收到的设置  TX TS 1/TS 2中的远程设备这  表示当前系数或预设值  从远程设备接收的数据被拒绝 |  |
| 二十九点二十四分 | 方向反馈 | RO | EP Ph 2/RC Ph 3：悬挂物方向改变反馈或预设馈送-  返回发送至  远程设备。位22，  EQPREVD，指示这是否是预设馈送-  返回或方向改变  反馈EP Ph 3/RC Ph 2：保留 |  |
| 三十一点半 | 均等阶段 | RO | 捕获00期间的均衡阶段  01：阶段1，10：阶段  2、11：第3 |  |

* + - 1. 均衡FIFO状态2寄存器i\_regf\_ip\_pl/i\_eq\_dbg\_fifo\_sts\_2，寄存器偏移：0x 10004 FC

**描述：**该寄存器包含均衡FIFO状态的高32位读取i\_eq\_dbg\_fifo\_sts\_1寄存器后，必须读取此寄存器

表14.29：均衡FIFO状态2寄存器

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 五比零 | 系数 | RO | 这只适用于Gen6。  相位0：存储从初始预设映射的初始本地TX系数。（厘米2）  第一阶段：保留。  EP Phase 2/RC Phase 3：存储远程设备的电流系数。（厘米2）  EP Phase 3/RC Phase 2：存储从远程设备接收的系数。 |  |
| 十一点六分 | 方向变化反馈 | RO | 方向变化反馈.这仅在Gen 6上有效。EP Phase 2/RC Phase 3：存储方向变化反馈（Gen 6最多6位）。EP阶段3/RC阶段2：保留 |  |

* + 1. LTSSM控制寄存器

默认情况下，LTSSM的值按照PCIe规范实现可以使用以下寄存器对这些超时值

* + - 1. LTSSM定时器控制0i\_regf\_ip\_pl/i\_ltssm\_tmr\_ctrl\_0，寄存器偏移：0x 1000560

表14.30：LTSSM定时器控制0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 十一比零 | LTSSM 1ms时间间隔 | RW | 该寄存器以（1024 ns）为单位保存1ms定时器间隔。可以调整该寄存器以改变LTSSM中的1ms超时。 | 12'd977 |
| 15点12分 | 保留 |  |  |  |
| 二十七点十六分 | LTSSM 2ms时间间隔 | RW | 该寄存器以（1024 ns）为单位保存2ms定时器间隔。可以调整该寄存器以改变LTSSM中的2ms超时。 | 1954年12月 |
| 28 | 保留 |  |  |  |

表14.30：LTSSM定时器控制0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 29 | LTSSM时间间隔缩放使能 | RW | 这是一个调试寄存器字段，用于启用LTSSM时间间隔缩放。  0：LTSSM时间间隔寄存器值以1024ns为单位。  1：LTSSM时间间隔值的单位为1024us。  注：将此位设置为1可启用扩展  时间间隔范围，是- yond PCIe规范定义的范围。因此，这只能用于调试目的，而不能用于功能模式。 | 0 |
| 30 | 禁用PHY LTSSM定时器 | RW | 禁用所有LTSSM计时器 | 0 |
| 31 | 启用快速链路训练 | RW | 用于通过缩减以下LTSSM定时器来加快链路训练：   * 在Detect.Quiet中，12ms定时器被缩减到120us。 * 在Detect.Active中，12ms定时器被缩减到120us。 | 0 |

* + - 1. LTSSM定时器控制1i\_regf\_ip\_pl/i\_ltssm\_tmr\_ctrl\_1，寄存器偏移：0x 1000564

表14.31：LTSSM定时器控制1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 十五比零 | LTSSM 4ms时间间隔 | RW | 该寄存器以（1024 ns）为单位保存4ms定时器间隔。此寄存器，TER可以调整到不同的LTSSM中的4ms超时。 | 16'd3907 |
| 三十一点十六分 | LTSSM 8ms时间间隔 | RW | 该寄存器以（1024 ns）为单位保存8 ms定时器间隔。此寄存器，TER可以调整到不同的LTSSM中的8ms超时。 | 16'd7813 |

* + - 1. LTSSM定时器控制2i\_regf\_ip\_pl/i\_ltssm\_tmr\_ctrl\_2，寄存器偏移：0x 1000568

表14.32：LTSSM定时器控制2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 十五比零 | LTSSM 12ms时间间隔 | RW | 该寄存器以（1024 ns）为单位保存12 ms定时器间隔。可以调整该寄存器以改变LTSSM中 | 16'd11719 |
| 三十一点十六分 | LTSSM 24ms时间间隔 | RW | 该寄存器以（1024 ns）为单位保存24 ms定时器间隔。可以调整该寄存器以改变LTSSM中 | 16'd23438 |

* + - 1. LTSSM定时器控制3i\_regf\_ip\_pl/i\_ltssm\_tmr\_ctrl\_3，寄存器偏移：0x 100056 C

表14.33：LTSSM定时器控制3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 十五比零 | LTSSM 32ms时间间隔 | RW | 该寄存器以（1024 ns）为单位保存32 ms定时器间隔。可以调整该寄存器以改变LTSSM中 | 16'd31250 |
| 三十一点十六分 | LTSSM 48ms时间间隔 | RW | 该寄存器以（1024 ns）为单位保存48 ms定时器间隔。可以调整该寄存器以改变LTSSM中 | 16'd46875 |

* + - 1. LTSSM定时器控制4i\_regf\_ip\_pl/i\_ltssm\_tmr\_ctrl\_4，寄存器偏移：0x 1000740

表14.34：LTSSM定时器控制4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 十五比零 | [LT64MSINT]  LTSSM 64ms时间间隔 | RW | 该寄存器以（1024 ns）为单位保存64 ms定时器间隔。可以调整该寄存器以改变LTSSM中 | 0xF424 |

* + 1. LTSSM转换冻结、中断控制状态寄存器

LTSSM可编程为在特定LTSSM转换事件后冻结LTSSM还可以编程为在特定LTSSM转换事件上断言中断这可以在以下寄存器中进行编程

* + - 1. i\_ltssm\_trans\_debug\_ctrl\_reg\_0i\_regf\_ip\_pl/i\_ltssm\_trans\_debug\_ctrl\_reg\_0，寄存器偏移：0x 10006 CC

该寄存器使固件能够对要检测的特定LTSSM状态转换进行编程，并可选择暂停以进行固件控制。

表14.35：i\_ltssm\_trans\_debug\_ctrl\_reg\_0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 六比零 | 当前LTSSM状态0 | RW | 当前LTSSM状态0。这是LTSSM  所需  以被检查并可选地暂停。请参阅7位LTSSM编码表的细节。 |  |
| 十四点八分 | 上一个LTSSM状态0 | RW | 上一个LTSSM状态0。这是LTSSM  所需  要检查。请参阅7位LTSSM编码表的细节。 |  |
|  |  |  |  |  |
| 二十二点十六分 | 当前LTSSM状态1 | RW | 当前LTSSM状态1。这是LTSSM  所需  以被检查并可选地暂停。请参阅7位LTSSM编码表的细节。 |  |
| 三十点二十四分 | 上一个LTSSM状态1 | RW | 上一个LTSSM状态1.这是LTSSM  所需  要检查。请参阅7位LTSSM编码表的细节。 |  |

* + - 1. i\_ltssm\_trans\_debug\_ctrl\_reg\_1i\_regf\_ip\_pl/i\_ltssm\_trans\_debug\_ctrl\_reg\_1，寄存器偏移：0x 10006 D 0

该寄存器使固件能够对要检测的特定LTSSM状态转换进行编程，并可选择暂停以进行固件控制。

表14.36：i\_ltssm\_trans\_debug\_ctrl\_reg\_1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 六比零 | 当前LTSSM状态2 | RW | 当前LTSSM状态2.这是LTSSM  所需  以被检查并可选地暂停。请参阅7位LTSSM编码表的细节。 |  |
| 十四点八分 | 上一个LTSSM状态2 | RW | 前LTSSM状态2.这是LTSSM  所需  要检查。请参阅7位LTSSM编码表的细节。 |  |
| 二十二点十六分 | 当前LTSSM状态3 | RW | 当前LTSSM状态3.这是LTSSM  所需  以被检查并可选地暂停。请参阅7位LTSSM编码表的细节。 |  |
| 三十点二十四分 | 上一个LTSSM状态3 | RW | 前LTSSM状态3.这是LTSSM  所需  要检查。请参阅7位LTSSM编码表的细节。 |  |

* + - 1. i\_ltssm\_trans\_debug\_ctrl\_reg\_2i\_regf\_ip\_pl/i\_ltssm\_trans\_debug\_ctrl\_reg\_2，寄存器偏移：0x 10006 D4

该寄存器使固件能够对要检测的特定LTSSM状态转换进行编程，并可选择暂停以进行固件控制。

表14.37：i\_ltssm\_trans\_debug\_ctrl\_reg\_2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 六比零 | 当前LTSSM状态4 | RW | 当前LTSSM状态4.这是LTSSM  所需  以被检查并可选地暂停。请参阅7位LTSSM编码表的细节。 |  |

表14.37：i\_ltssm\_trans\_debug\_ctrl\_reg\_2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 十四点八分 | 前LTSSM状态4 | RW | 前LTSSM状态4.这是LTSSM  所需  要检查。请参阅7位LTSSM编码表的细节。 |  |
| 二十二点十六分 | 当前LTSSM状态5 | RW | 当前LTSSM状态5.这是LTSSM  所需  以被检查并可选地暂停。请参阅7位LTSSM编码表的细节。 |  |
| 三十点二十四分 | 前LTSSM状态5 | RW | 前LTSSM状态5.这是LTSSM  所需  要检查。请参阅7位LTSSM编码表的细节。 |  |

* + - 1. i\_ltssm\_trans\_debug\_ctrl\_reg\_3i\_regf\_ip\_pl/i\_ltssm\_trans\_debug\_ctrl\_reg\_3，寄存器偏移：0x 10006 D8

该寄存器使固件能够对要检测的特定LTSSM状态转换进行编程，并可选择暂停以进行固件控制。

表14.38：i\_ltssm\_trans\_debug\_ctrl\_reg\_3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 六比零 | 当前LTSSM状态6 | RW | 当前LTSSM状态6。这是LTSSM  所需  以被检查并可选地暂停。请参阅7位LTSSM编码表的细节。 |  |
| 十四点八分 | 前LTSSM状态6 | RW | 前LTSSM状态6.这是LTSSM  所需  要检查。请参阅7位LTSSM编码表的细节。 |  |

表14.38：i\_ltssm\_trans\_debug\_ctrl\_reg\_3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 二十二点十六分 | 当前LTSSM状态7 | RW | 当前LTSSM状态7.这是LTSSM  所需  以被检查并可选地暂停。请参阅7位LTSSM编码表的细节。 |  |
| 三十点二十四分 | 前LTSSM状态7 | RW | 前LTSSM状态7.这是LTSSM  所需  要检查。请参阅7位LTSSM编码表的细节。 |  |

* + - 1. i\_ltssm\_transs\_debug\_control\_status\_regi\_regf\_ip\_pl/i\_ltssm\_transs\_debug\_control\_status\_reg，寄存器偏移：0x 10006 DC

该寄存器包括8个LTSSM状态转换的ltssm转换控制和状态，可编程用于固件监视和控制。

表14.39：i\_ltssm\_transs\_debug\_control\_status\_reg

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 七比零 | 无冻结状态的LTSSM状态 | RW1Clr | 无冻结状态的LTSSM状态此字段包含1位  8个国家中的每一个的状况  可编程LTSSM状态转换。  1：表示发生了编程的ltssm转换，并且未编程为在该转换时冻结。  0：表示编程的ltssm转换未发生或编程为冻结。  写入1以清除 |  |

表14.39：i\_ltssm\_transs\_debug\_control\_status\_reg

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 十五点八分 | 具有冻结状态的LTSSM状态 | RW1Clr | 具有冻结状态的LTSSM状态  该字段包含8个可编程LTSSM状态转换的每个状态- tus 1位。  1：表示发生了编程的ltssm转换，并且编程为在该转换上冻结。LTSSM  冻结在当前LTSSM状态，等待固件干预解除冻结。  0：表示编程的ltssm转换尚未发生或未编程为冻结。  写1为清除。 |  |
| 二十三点十六分 | LTSSM状态转换冻结使能。 | RW | LTSSM状态转换冻结使能。  固件可使用此字段在编程的LTSSM转换后冻结LTSSM  发生i\_ltssm\_transs\_debug\_status\_reg  每个位用于启用相应的LTSSM状态转换，  i\_ltssm\_trans\_debug\_status\_reg.  1：LTSSM状态将被冻结后，programmed过渡ocurs. LTSSM将保持在current\_state，通过丢弃所有正在接收的操作系统。  0：LTSSM状态将不会冻结后，编程过渡。  LTSSM将继续到下一个合法状态。 |  |

表14.39：i\_ltssm\_transs\_debug\_control\_status\_reg

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特位置 | 寄存器字段 | Attr | 描述 | 复位 |
| 三十一点二十四分 | LTSSM状态转换检查使能。 | RW | LTSSM状态转换检查使能。此位使能LTSSM转换检查。  此字段启用LTSSM转换  检查在i\_ltssm\_transs\_debug\_status\_reg中编程。每个位用于使能相应的LTSSM  i\_ltssm\_transs\_debug\_status\_reg中的状态转换  1：LTSSM状态转换检查已启用。0：LTSSM状态转换  已禁用  注：当LTSSM状态转换1检查启用==1时，如果发生编程转换且LTSSM状态转换  LTSSTTRANM位中未屏蔽UART |  |

* + 1. LTSSM转换原因状态寄存器

控制器在状态寄存器中反映每个LTSSM转换的详细信息以下寄存器可用于检查LTSSM转换事件的描述。

* + - 1. i\_regf\_ip\_pl/i\_ltssm\_transs\_cause\_status\_reg寄存器偏移：0x 10006 E0

该寄存器反映最近的LTSSM状态转换和转换原因。

表14.40：i\_regf\_ip\_pl/i\_ltssm\_transs\_cause\_status\_reg

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 比特 | 接入 | 复位值 | 描述 |
| LTSSM状态转换原因  [LTSTTRC] | 二十三点十六分 | R | 硬：0x00 | LTSSM状态转换原因。  此字段包含最近LTSSM转换原因的8位编码值。 |
| 当前LTSSM状态[CRLTST] | 十四点八分 | R | 硬：0x00 | 当前LTSSM状态。  此字段包含最近LTSSM转换的7位当前LTSSM状态 |
| 上一个LTSSM状态[PRLTST] | 六比零 | R | 硬：0x00 | 上一个LTSSM状态。  此字段包含最近LTSSM转换的7位先前LTSSM状态 |

* + - 1. i\_ltssm\_transition\_cause\_enable\_[n]i\_regf\_ip\_pl/i\_ltssm\_transition\_cause\_enable\_0，寄存器偏移：0x 1000750描述：LTSSM转换原因使能寄存器0

某些LTSSM转换可能不适用于调试目的。这种转换可以被屏蔽而不被记录。此寄存器控制在i\_ltssm\_transs\_cause\_status\_reg寄存器中记录LTSSM转换原因。控制器中有大约160个内部LTSSM状态此寄存器为每个LTSSM状态提供1位，以使能/禁用i\_ltssm\_transs\_cause\_status\_reg中的报告

表14.41：i\_ltssm\_transition\_cause\_enable\_[n]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 比特 | 接入 | 复位值 | 描述 |
| [LTTRCSEN0]  LTSSM转换原因报告启用 | 三十一比零 | RW | 0xFFFFFFFF | [LTTRCSEN0]  LTSSM状态0到31的LTSSM转换原因报告启用  该寄存器包含每个LTSSM状态的一个位，用于在进入该状态时记录LTSSM转换原因  i\_ltssm\_transcause\_status\_reg寄存器中的LTSSM状态。  1：在进入此状态时启用LTSSM转换原因日志记录  0：进入此状态时未启用LTSSM转换原因日志记录 |

其他LTSSM国家的比特在以下国家实施：

**LTSSM状态[63：32]：i\_regf\_ip\_pl/i\_ltssm\_transition\_cause\_enable\_1**，寄存器偏移：0x 1000754**LTSSM状态[95：64]：i\_regf\_ip\_pl/i\_ltssm\_transition\_cause\_enable\_2**，寄存器偏移：0x 1000758**LTSSM状态[127：96]：i\_regf\_ip\_pl/i\_ltssm\_transition\_cause\_enable\_3**，寄存器偏移：0x 100075 C**LTSSM状态[159：128]：i\_regf\_ip\_pl/i\_ltssm\_transition\_cause\_enable\_4**，寄存器偏移：0x 1000760

* + - 1. i\_ltssm\_transs\_cause\_fifo\_data\_outi\_regf\_ip\_pl/i\_ltssm\_transs\_cause\_fifo\_data\_out，寄存器偏移：0x 1000764

描述：控制器在内部FIFO中记录LTSSM转换事件。通过读取该寄存器可以访问该FIFO。最多64个LTSSM状态转换和转换原因记录在此FIFO中。必须多次读取该寄存器以获取记录的条目。

表14.42：i\_ltssm\_transcause\_fifo\_data\_out

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 比特 | 接入 | 复位值 | 描述 |
| LTSSM传输原因FI- FO数据输出 | 三十一比零 | RO | 0 | 位[23：16]-LTSSM状态转换原因位[14：8] -当前LTSSM状态  位[6：0] -上一个LTSSM状态 |

* + - 1. i\_ltssm\_transs\_cause\_fifo\_cyclei\_regf\_ip\_pl/i\_ltssm\_transs\_cause\_fifo\_cycle，寄存器偏移：0x 100076 C

描述：此寄存器用于控制LTSSM转换原因调试FIFO行为。

表14.43：i\_ltssm\_transcause\_fifo\_codeword

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 比特 | 接入 | 复位值 | 描述 |
| LTSSM转换原因FIFO控制  [LTCFC] | 0 | RW | 0 | 1：存储前64个LTSSM转换，  0：存储最近的64个LTSSM转换。 |

7-位LTSSM状态编码

LTSSM寄存器中使用的7位LTSSM状态编码定义如下：

表14.44：7位LTSSM状态编码

|  |  |
| --- | --- |
| 检测\_安静 | 7'd0 |
| 检测安静进入 | 7'd1 |
| 检测\_活动 | 7'd2 |
| 检测\_ACTIVE\_1 | 7'd3 |
| 检测\_ACTIVE\_2 | 7'd4 |
| 检测\_ACTIVE\_3 | 7'd5 |
| RCVR\_DETECTED\_ST | 7'd6 |
| RCVR\_DETECTED\_1 | 7'd7 |
| 轮询\_活动 | 7'd8 |
| 轮询\_ACTIVE\_1 | 7'd9 |
| 轮询\_ACTIVE\_2 | 7'd10 |
| 轮询\_ACTIVE\_3 | 7'd11 |
| 轮询\_依从性 | 7'd12 |
| 投票\_COMPANCE\_1 | 7'd13 |
| POLLING\_CONFIGConfig\_LW | 7'd14 |
| POLLING\_CONFIG\_1 | 7'd15 |
| POLLING\_CONFIG\_2 | 7'd16 |
| CONFIG\_LW\_START\_RC | 7'd17 |
| CONFIG\_LW\_START\_RC\_1 | 7'd18 |
| CONFIG\_LW\_START\_RC\_2 | 7'd19 |
| 配置\_LW\_ACC\_RC | 7'd20 |
| CONFIG\_LANENUM\_WAIT\_RC | 7'd21 |
| CONFIG\_LANENUM\_WAIT\_RC\_1 | 7'd22 |
| CONFIG\_LANENUM\_ACC\_RC | 7'd23 |
| CONFIG\_LW\_START\_EP | 7'd24 |
| CONFIG\_LW\_START\_EP\_1 | 7'd25 |
| CONFIG\_LW\_START\_EP\_2 | 7'd26 |
| CONFIG\_LW\_ACC\_EP | 7'd27 |
| CONFIG\_LANENUM\_WAIT\_EP | 7'd28 |
| CONFIG\_LANENUM\_WAIT\_EP\_1 | 7'd29 |
| CONFIG\_LANENUM\_ACC\_EP | 7'd30 |
| CONFIG\_LANENUM\_ACC\_EP\_1 | 7'd31 |
| 虚拟状态1 | 7'd32 |
| 配置\_完成 | 7'd33 |
| CONFIG\_COMPLETE\_1 | 7'd34 |
| CONFIG\_COMPLETE\_2 | 7'd35 |
| CONFIG\_IDLE | 7'd36 |

表14.44：7位LTSSM状态编码

|  |  |
| --- | --- |
| CONFIG\_IDLE\_1 | 7'd37 |
| 虚拟状态2 | 7'd38 |
| 虚拟状态3 | 7'd39 |
| 虚拟状态\_4 | 7'd40 |
| L0状态 | 7'd41 |
| 恢复\_RCVR\_恢复 | 7'd42 |
| RECOVERY\_RCVR\_LOCK\_1 | 7'd43 |
| 恢复\_RCVR\_CFG | 7'd44 |
| RECOVERY\_RCVR\_CFG\_1 | 7'd45 |
| 恢复\_空闲 | 7'd46 |
| RECOVERY\_IDLE\_1 | 7'd47 |
| DISABLE\_LINK | 7'd48 |
| DISABLE\_LINK\_1 | 7'd49 |
| DISABLE\_LINK\_2 | 7'd50 |
| DISABLE\_LINK\_3 | 7'd51 |
| DISABLE\_LINK\_4 | 7'd52 |
| DISABLE\_LINK\_5 | 7'd53 |
| DISABLE\_LINK\_6 | 7'd54 |
| DISABLE\_LINK\_7 | 7'd55 |
| 热门文章 | 7'd56 |
| HOT\_RESET\_1 | 7'd57 |
| HOT\_RESET\_2 | 7'd58 |
| HOT\_RESET\_3 | 7'd59 |
| L0S\_入口 | 7'd60 |
| L0S\_1 | 7'd61 |
| L0S\_2 | 7'd62 |
| L0S\_3 | 7'd63 |
| L0S\_4 | 7'd64 |
| L0S\_5 | 7'd65 |
| 等待链接德克萨斯州 | 7'd66 |
| TX\_FTS\_输入 | 7'd67 |
| TX\_FTS\_1 | 7'd68 |
| TX\_FTS\_2 | 7'd69 |
| TX\_ELEC\_IDLE\_ST | 7'd70 |
| TX\_ELEC\_IDLE\_1 | 7'd71 |
| TX\_ELEC\_IDLE\_2 | 7'd72 |
| TX\_ELEC\_IDLE\_3 | 7'd73 |
| 恢复速度 | 7'd74 |
| RECOVERY\_SPEED\_1 | 7'd75 |

表14.44：7位LTSSM状态编码

|  |  |
| --- | --- |
| RECOVERY\_SPEED\_2 | 7'd76 |
| RECOVERY\_SPEED\_3 | 7'd77 |
| POLLING\_COMPANCE\_GEN23 | 7'd78 |
| PolLING\_COMPANCE\_GEN23\_1 | 7'd79 |
| PolLING\_COMPANCE\_GEN23\_2 | 7'd80 |
| PolLING\_COMPANCE\_GEN23\_3 | 7'd81 |
| PolLING\_COMPANCE\_GEN23\_4 | 7'd82 |
| PolLING\_COMPANCE\_GEN23\_5 | 7'd83 |
| PolLING\_COMPANCE\_GEN23\_6 | 7'd84 |
| PolLING\_COMPANCE\_GEN23\_7 | 7'd85 |
| PolLING\_COMPANCE\_GEN23\_8 | 7 'd86 |
| LOOPBACK\_Slave\_ENTRY | 7'd87 |
| LOOPBACK\_SLAVE\_ENTRY\_FROM\_RECOVERY | 7'd88 |
| LOOPP\_SLAVE\_EXIT\_1 | 7 'd89 |
| LOOPBACK\_SLAVE\_EXIT | 7'd90 |
| 循环\_SLAVE\_GEN2\_1 | 7'd91 |
| 循环\_SLAVE\_GEN2\_2 | 7'd92 |
| 循环\_SLAVE\_GEN2\_3 | 7'd93 |
| 循环\_SLAVE\_GEN2\_4 | 7'd94 |
| 循环\_SLAVE\_GEN2\_5 | 7'd95 |
| 回送从机活动 | 7'd96 |
| L1入口 | 7'd97 |
| L1\_1 | 7'd98 |
| L1\_2 | 7'd99 |
| L1\_3 | 7'd100 |
| L1\_4 | 7'd101 |
| L1\_IDLE | 7'd102 |
| L1\_退出 | 7 'd103 |
| L2入口 | 7'd104 |
| L2\_1 | 7'd105 |
| L2\_2 | 7'd106 |
| L2\_3 | 7'd107 |
| L2\_4 | 7'd108 |
| L2\_5 | 7'd109 |
| L2\_IDLE | 7'd110 |
| LOOPBACK\_MASTER\_ENTRY | 7'd111 |
| 循环主条目1 | 7'd112 |
| 循环主条目2 | 7'd113 |
| 循环主条目3 | 7'd114 |
| 循环主条目4 | 7'd115 |

表14.44：7位LTSSM状态编码

|  |  |
| --- | --- |
| 循环主条目5 | 7'd116 |
| LOOPBACK\_MASTER\_ENTRY\_FROM\_RECOVERY | 7'd117 |
| LOOPBACK\_MASTER\_ACTIVE | 7'd118 |
| LOOPBACK\_MASTER\_EXIT | 7'd119 |
| LOOPP\_MASTER\_EXIT\_1 | 7'd120 |
| LOOPP\_MASTER\_EXIT\_2 | 7'd121 |
| 恢复均衡阶段0 | 7'd122 |
| 恢复均衡阶段1 | 7 'd123 |
| 恢复均衡阶段2\_1 | 7'd124 |
| 恢复均衡阶段2\_2 | 7'd125 |
| 恢复均衡阶段3\_1 | 7'd126 |
| 恢复均衡阶段3\_2 | 7'd127 |

* + 1. L0->恢复LTSSM状态转换原因编码

L0->Recovery转换的LTSSM状态转换原因如下所示

表14.45：L0->恢复LTSSM状态转换原因编码

|  |  |
| --- | --- |
| LTSSM状态转换原因[7：0] | 描述 |
| 8'd0 | RP重新训练链接。仅在RP模式下有效 |
| 8'd1 | EP链接速度变化再培训链接。仅在EP模式下有效 |
| 8'd2 | EP 8 GT/s请求EQ重新训练链接。仅在EP模式下有效 |
| 8'd3 | EP 16 GT/s请求EQ重新训练链接。仅在EP模式下有效 |
| 8'd4 | EP 32 GT/s请求EQ重新训练链接。仅在EP模式下有效 |
| 8'd255 | EP 64 GT/s请求EQ重新训练链接。仅在EP模式下有效 |
| 8'd5 | RP链路禁用重新训练。仅在RP模式下有效 |
| 8'd6 | RP热重置重新训练。仅在RP模式下有效 |
| 8'd7 | RP环回主再训练。仅在RP模式下有效 |
| 8'd8 | NFTS在L0s期间停止。 |
| 8'd9 | RP自主均衡速度变化再训练链路。仅在RP模式下有效 |
| 8'd10 | RP Autonomous Gen1至Gen2变速再培训链路。仅在RP模式下有效 |
| 8'd11 | L0->L1条目。 |
| 8'd12 | L0->L2入口。 |
| 8'd13 | Pipe\_rxelecidle已断言，但未收到EIOS。 |
| 8'd14 | 在数据流期间取消断言Pipe\_rxvalid。 |
| 8'd15 | 在数据流期间接收到TS 1/TS 2。 |
| 8'd16 | DL链路重训练请求被断言。 |
| 8'd17 | 本地检测到帧错误 |
| 8'd18 | FC重新训练链接。 |
| 8'd19 | 链路升级配置重新训练由本地固件启动的链路 |
| 8'd20 | L0->Tx.L0s条目 |

表14.45：L0->恢复LTSSM状态转换原因编码

|  |  |
| --- | --- |
| LTSSM状态转换原因[7：0] | 描述 |
| 8'd21 | 控制器固件发起IORecal重新训练请求。 |
| 8'd22 | PHY发起IORecal请求。 |

### LTSSM转换编码表14.46：检测

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No. | 先前状态 | 当前状态 | 跨因 | 转型原因 |
| 1 | 检测\_安静 | 检测\_安静 | 8'd0 | 启用未编程的 |
| 2 | 检测\_活动 | 8'd1 | 任何车道 |
| 3 | 8'd2 | 在任何车道上，100 us等待计时器的电气怠速中断 |
| 4 | 8'd3 | 任何通道上的电气怠速中断，等待计时器为1 ms |
| 5 | 8'd4 | 任何通道上的电气怠速中断，等待计时器为2 ms |
| 6 | 8'd5 | 12 ms超时后（如果编程了启用快速链路训练，则该值将为120 us |
| 7 | 检测安静进入 | 检测\_安静 | 8'd0 | PHY状态已取消断言。 |
| 8 | 8'd1 | PHY已完成到P1 |
| 9 | 8'd2 | 12 ms超时后。 |
| 10 | 检测\_活动 | 检测\_ACTIVE\_1 | 8'd0 | 收到PHY状态 |
| 11 | 检测\_ACTIVE\_1 | RCVR\_DETECTED\_ST | 8'd0 | 已在所有通道上检测到接收器 |
| 12 | 检测\_ACTIVE\_2 | 8'd1 | 在某些通道上检测到接收器。 |
| 13 | 检测安静进入 | 8'd2 | 未检测到接收器。 |
| 14 | 8'd3 | 12 ms超时后。 |
| 15 | 检测\_ACTIVE\_2 | 检测\_ACTIVE\_3 | 8'd0 | 12 ms超时后重新启动接收器检测序列。（如果编程了启用快速链接训练，超时将为120 us |
| 16 | 检测\_ACTIVE\_3 | RCVR\_DETECTED\_ST | 8'd0 | 在与先前状态相同的通道上检测到接收器。 |
| 17 | 检测安静进入 | 8'd1 | 在不同通道上检测到接收器为先前状态。 |
| 18 | 检测安静进入 | 8'd2 | 未检测到接收器，12 ms超时过期。 |
| 19 | RCVR\_DETECTED\_ST | RCVR\_DETECTED\_1 | 8'd0 | 始终在将PHY状态设置为P0后执行。 |
| 20 | RCVR\_DETECTED\_1 | 轮询\_活动 | 8'd0 | 一旦接收到PHY状态，则始终执行此操作。 |

表14.47：投票

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No. | 先前状态 | 当前状态 | 跨因 | 转型原因 |
| 1 | 轮询\_活动 | 轮询\_依从性 | 8'd0 | Reg输入合规性设置，不支持gen23 |
| 2 | 轮询\_依从性  \_GEN23 | 8'd1 | 注册输入合规设置和gen23功能 |
| 3 | 轮询\_ACTIVE\_1 | 8'd2 | 未设置 |

表14.47：投票

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No. | 先前状态 | 当前状态 | 跨因 | 转型原因 |
| 4 | 轮询\_ACTIVE\_1 | 轮询配置 | 8'd0 | 在检测期间检测到接收器的任何通道接收到满足以下条件中的任一个的八个连续训练序列（或其补充）：   1. 通道号和链路号设置为PAD且合规接收位（符号5的位4）为0b的TS 1。 2. 通道和链路号设置为PAD且环回位（符号5的位2）为 1b的TS 1。 3. 接收到一个TS 1或TS 2有序集后，将传输通道和链路号设置为PAD的TS 2，并且至少传输1024个TS 1   在Detect期间检测到接收器的至少一组预定通道自从进入Polling.Active.以来已经检测到至少一次从电空闲退出。注意：这可以防止一个或多个坏的接收器或发送器阻止配置有效的链路，并允许在轮询中进行广告培训。预定通道的确切集合是特定于实现的。  注意事项：   * + 直到[PCIe-1.1]，该预定集合等于检测到接收器的通道的总集合。   + 接收八个连续TS1或TS2有序集的任何通道应在进入Polling.Active后至少检测到一次从电气空闲退出。 |

表14.47：投票

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No. | 先前状态 | 当前状态 | 跨因 | 转型原因 |
| 5 | 轮询\_ACTIVE\_1 | 轮询\_依从性 | 8'd1 | 如果（a）或（b）为真：a。不是来自上述（ii）的预定通道组中的所有通道都检测到从进入轮询.活动后退出电空闲。B.在检测期间检测到接收器的任何通道接收八个连续的TS1有序集（或其补码），其中通道和链路号被设置为PAD，合规性+F7接收位（符号5的位4）是1b，并且  环回位（符号5的位2）为0b。注：如果在所有通道上应用被动测试负载，则设备将进入轮询.合规性。实施检查检测到Rx的一个或多个通道是否仍处于电气空闲状态进入轮询。 |
| 6 | 轮询\_ACTIVE\_1 | POLLING\_COMPANCE\_GEN23 | 8'd2 | 如果（a）或（b）为真：a。不是来自上述（ii）的预定通道组中的所有通道都检测到从进入轮询.活动后退出电空闲。B.在检测期间检测到接收器的任何通道接收八个连续的TS1有序集（或其补码），其中通道和链路号被设置为PAD，合规性+F7接收位（符号5的位4）是1b，并且  环回位（符号5的位2）为0b。注：如果在所有通道上应用被动测试负载，则设备将进入轮询.合规性。实施检查检测到Rx的一个或多个通道是否仍处于电气空闲状态进入轮询。 |
| 7 | 轮询\_ACTIVE\_1 | 8'd3 | 任何8个TS1与合规接收 |
| 8 | 轮询\_ACTIVE\_1 | TX\_ELEC\_IDLE\_ST | 8'd4 | 如果未收到具有合规性设置的 |
| 9 | 轮询\_ACTIVE\_1 | 轮询配置 | 8'd5 | 在所有活动信道上发送1024个TS 1有序集 |
| 10 | 轮询\_依从性 | 轮询\_活动 | 8'd0 | 如果车道从非空闲状态转换，则 |
| 11 | POLLING\_COMPANCE\_GEN23 | 轮询\_依从性  \_GEN23\_1 | 8'd0 | EIOS将在符合性模式之前传输，因此首先 |
| 12 | 轮询\_依从性  \_GEN23\_4 | 8'd1 | 第1代和TX\_COMPANCE\_1时的tx\_state |
| 13 | 轮询\_依从性  \_GEN23\_2 | 8'd2 | 在遵从性传输之前无需发送IDL有序集 |

表14.47：投票

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No. | 先前状态 | 当前状态 | 跨因 | 转型原因 |
| 14 | PolLING\_COMPANCE\_GEN23\_1 | 轮询\_依从性  \_GEN23\_2 | 8'd0 | 发送第二个IDL序列 |
| 15 | PolLING\_COMPANCE\_GEN23\_2 | 轮询\_依从性  \_GEN23\_3 | 8'd0 | 驱动SERDES至电气怠速 |
| 16 | PolLING\_COMPANCE\_GEN23\_3 | 轮询\_依从性  \_GEN23\_4 | 8'd0 | 等待2 ms超时后发送合规性模式或修改后的合规性模式（tx\_state = TX\_COMPANCE\_1） |
| 17 | PolLING\_COMPANCE\_GEN23\_4 | 轮询\_依从性  \_GEN23\_5 | 8'd0 | 一个或多个车道退出电气怠速 |
| 18 | 8'd1 | 链路控制2寄存器位从1变为0（输入合规性位） |
| 19 | 8'd2 |  |
| 20 | PolLING\_COMPANCE\_GEN23\_5 | 轮询\_活动 | 8'd0 | 如果当前在5G下运行，或者由于设置了链路控制2寄存器的Enter Compliance位而进入Polling. Compliance，则必须在返回Polling. Active之前发送8个EIOS序列。 |
| 21 | 轮询\_依从性  \_GEN23\_6 | 8'd1 | 如果链路控制2寄存器的一致性位仍为1 |
| 22 | PolLING\_COMPANCE\_GEN23\_6 | 轮询\_依从性  \_GEN23\_7 | 8'd0 | 发送6个空闲序列时钟 |
| 23 | PolLING\_COMPANCE\_GEN23\_7 | 轮询\_依从性  \_GEN23\_8 | 8'd0 | 驱动SERDES至电气怠速 |
| 24 | PolLING\_COMPANCE\_GEN23\_8 | 轮询\_活动 | 8'd0 | 设置速度等待8次时钟后，在tx\_start\_idle\_reg之后，更改current\_speed之前，需要使用variable\_core\_start\_en，以确保pipe\_rate不会在pipe\_tx\_start\_idle之前更改 |
| 25 | 轮询配置 | POLLING\_CONFIG\_1 | 8'd0 | 一直做 |
| 26 | POLLING\_CONFIG\_1 | POLLING\_CONFIG\_2 | 8'd0 | 收到有效的TS2 |
| 27 | TX\_ELEC\_IDLE\_ST | 8'd1 | 未收到无错误TS2，48 ms超时计数器过期 |
| 28 | POLLING\_CONFIG\_2 | CONFIG\_LW\_START\_RC | 8'd0 | 等待直到在检测到接收器的任何通道上接收到8个连续的无差错TS2有序集，并且发送16个TS2有序集。当端口为RP时 |
| 29 | CONFIG\_LW\_START\_EP | 8'd1 | 等待直到在检测到接收器的任何通道上接收到8个连续的无差错TS2有序集，并且发送16个TS2有序集。当端口为EP时 |
| 30 | TX\_ELEC\_IDLE\_ST | 8'd2 | 8在检测到的通道上未接收到无错误TS2，48 ms超时计数器过期 |

表14.48：配置

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No. | 先前状态 | 当前状态 | 跨因 | 转型原因 |
| 1 | CONFIG\_LW\_START\_RC | DISABLE\_LINK | 8'd0 | 已收到链路禁用请求，或已断言禁用所有链路 |
| 2 | LOOPBACK\_MASTER\_ENTRY | 8'd1 | 如果下游端口和网络请求 |
| 3 | LOOPBACK\_Slave\_ENTRY | 8'd2 | 如果请求 |
| 4 | LOOPBACK\_Slave\_ENTRY | 8'd3 | 下一个状态是环回，如果任何一个通道是传输TS 1有序集接收到两个consecu- tive TS 1有序集与环回位作为- serted和增强链路行为控制位设置为01 b。 |
| 5 | CONFIG\_LW\_START\_RC\_1 | 8'd4 | 如果未设置或未接收到 |
| 6 | CONFIG\_LW\_START\_RC\_1 | DISABLE\_LINK | 8'd0 | 已收到链路禁用请求，或已断言禁用所有链路 |
| 7 | LOOPBACK\_MASTER\_ENTRY | 8'd1 | 如果下游端口和网络请求 |
| 8 | LOOPBACK\_Slave\_ENTRY | 8'd2 | 如果请求 |
| 9 | 8'd3 | 下一个状态是环回，如果任何一个通道是传输TS 1有序集接收到两个consecu- tive TS 1有序集与环回位作为- serted和增强链路行为控制位设置为01 b。 |
| 10 | CONFIG\_LW\_START\_RC\_2 | 8'd4 | 第2个TS1与PAD一起 |
| 11 | TX\_ELEC\_IDLE\_ST | 8'd5 | 未收到第二个TS1（带PAD），24 ms超时计数器过期 |
| 12 | CONFIG\_LW\_START\_RC\_2 | 8'd6 | 在任何车道上查找要接收的PAD/PAD链路/车道号 |
| 13 | CONFIG\_LW\_START\_EP | 8'd7 | 启用交叉链接，交叉链接计时器超时 |
| 14 | TX\_ELEC\_IDLE\_ST | 8'd8 | 24 ms超时计数器过期 |
| 15 | CONFIG\_LW\_START\_RC\_2 | LOOPBACK\_Slave\_ENTRY | 8'd0 | 如果在所有活动通道上接收到的两个连续TS1帧中发现设置了1位，则进入LOOPBACK\_SLAVE状态。 |
| 16 | 8'd1 | 下一个状态是环回，如果任何一个通道是传输TS 1有序集接收到两个consecu- tive TS 1有序集与环回位作为- serted和增强链路行为控制位设置为01 b。 |
| 17 | 配置\_LW\_ACC\_RC | 8'd2 | 如果在TS有序集中有 |
| 18 | TX\_ELEC\_IDLE\_ST | 8'd3 | 24 ms超时计数器过期 |
| 19 | 配置\_LW\_ACC\_RC | CONFIG\_LANENUM\_WAIT\_RC | 8'd0 | 等待4个TS周期，以允许在所有良好通道上接收链路ID。分配通道号如果tx\_state = TX\_TS\_ID\_LAST |
| 20 | TX\_ELEC\_IDLE\_ST | 8'd1 | 2个TS1接收到PAD或24 ms超时计数器过期 |
| 21 | CONFIG\_LANENUM  \_等待\_RC | CONFIG\_LANENUM\_WAIT\_RC\_1 | 8'd0 | 总是过渡 |

表14.48：配置

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No. | 先前状态 | 当前状态 | 跨因 | 转型原因 |
| 22 | CONFIG\_LANENUM  \_WAIT\_RC\_1 | CONFIG\_LANENUM\_ACC\_RC | 8'd0 | 接收到TS有序集，传输的通道号没有变化（使用修改后的TS时） |
| 23 | TX\_ELEC\_IDLE\_ST | 8'd1 | 2个TS1收到PAD或24 ms超时计数器过期（当使用修改的TS时） |
| 24 | CONFIG\_LANENUM\_ACC\_RC | 8'd2 | 接收到TS有序集，传输的通道号 |
| 25 | TX\_ELEC\_IDLE\_ST | 8'd3 | 2个TS1接收到PAD或24 ms超时计数器过期 |
| 26 | CONFIG\_LANENUM  联系我们 | 配置\_完成 | 8'd0 | 在进入CONFIG COMPUTER（配置完成）之前，等待APN端口设置为正确的TS2值（使用修改后的TS） |
| 27 | CONFIG\_LANENUM\_WAIT\_RC | 8'd1 | 所有活动通道上的链路ID和通道号不匹配（使用修改后的TS） |
| 28 | TX\_ELEC\_IDLE\_ST | 8'd2 | 2个TS 1接收到PAD或24 ms超时计数器过期（使用修改后的TS） |
| 29 | 配置\_完成 | 8'd3 | 链接ID和车道号都匹配。 |
| 30 | CONFIG\_LANENUM\_WAIT\_RC | 8'd4 | 所有活动通道 |
| 31 | TX\_ELEC\_IDLE\_ST | 8'd5 | 2个TS1接收到PAD或24 ms超时计数器过期 |
| 27 | CONFIG\_LW\_START\_EP | DISABLE\_LINK | 8'd0 | Rx断言禁用链路 |
| 28 | LOOPBACK\_Slave\_ENTRY | 8'd1 | 下一个状态是环回，如果所有的通道是transmitting的TS1有序集，这也是接收的TS1有序集，接收两个连续的TS1或有序集断言的环回位环回位。TS1有序集 |
| 29 | 8'd2 | 下一个状态是环回，如果任何一个通道是传输TS 1有序集接收两个consecu- tive TS 1有序集与环回位作为- serted和增强链路行为控制位设置为01 b。 |
| 30 | CONFIG\_LW\_START\_EP\_1 | 8'd3 | 如果未设置或未接收到 |
| 31 | 配置\_LW  \_START\_EP\_1 | LOOPBACK\_Slave\_ENTRY | 8'd0 | 如果在所有活动通道上接收到的两个连续TS1帧中发现设置了1位，则进入LOOPBACK\_SLAVE状态。 |
| 32 | 8'd1 | 下一个状态是环回，如果任何一个通道是传输TS 1有序集接收两个consecu- tive TS 1有序集与环回位作为- serted和增强链路行为控制位设置为01 b。 |
| 33 | DISABLE\_LINK | 8'd2 | Rx断言禁用链路 |
| 34 | CONFIG\_LW\_START\_EP\_2 | 8'd3 | 在任何车道上查找要接收的有效非PAD链路编号 |
| 35 | CONFIG\_LW\_START\_RC | 8'd4 | 如果交叉链接启用且交叉链接定时器超时 |
| 36 | TX\_ELEC\_IDLE\_ST | 8'd5 | 24 ms超时计数器过期 |

表14.48：配置

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No. | 先前状态 | 当前状态 | 跨因 | 转型原因 |
| 37 | CONFIG\_LW\_START\_EP\_2 | LOOPBACK\_Slave\_ENTRY | 8'd0 | 如果在所有活动通道上接收到的两个连续TS1帧中发现设置了1位，则进入LOOPBACK\_SLAVE状态。 |
| 38 | 8'd1 | 下一个状态是环回，如果任何一个通道是传输TS 1有序集接收两个consecu- tive TS 1有序集与环回位作为- serted和增强链路行为控制位设置为01 b。 |
| 39 | DISABLE\_LINK | 8'd2 | Rx断言禁用链路 |
| 40 | CONFIG\_LW\_ACC\_EP | 8'd3 | 链路ID已配置 |
| 41 | TX\_ELEC\_IDLE\_ST | 8'd4 | 24 ms超时计数器过期 |
| 42 | CONFIG\_LW\_ACC\_EP | CONFIG\_LANENUM\_WAIT\_EP | 8'd0 | 等待Last TX TS状态开始发送MTS 1 |
| 43 | CONFIG\_LANENUM\_WAIT\_EP | 8'd1 | 如果不使用 |
| 44 | TX\_ELEC\_IDLE\_ST | 8'd2 | 2个TS1接收到PAD或24 ms超时计数器过期 |
| 45 | CONFIG\_LANENUM  \_等待\_EP | CONFIG\_LANENUM\_WAIT\_EP\_1 | 8'd0 | 总是过渡 |
| 46 | CONFIG\_LANENUM  \_WAIT\_EP\_1 | CONFIG\_LANENUM\_ACC\_EP | 8'd0 | 如果任何通道接收两个连续的TS2有序集（使用修改的TS） |
| 47 | TX\_ELEC\_IDLE\_ST | 8'd1 | 2个TS 1接收到PAD或2 ms超时计数器过期（使用修改后的TS） |
| 48 | CONFIG\_LANENUM\_ACC\_EP | 8'd2 | 接收到TS有序集，传输的通道号 |
| 49 | TX\_ELEC\_IDLE\_ST | 8'd3 | 2个TS1接收到PAD或2 ms超时计数器过期 |
| 50 | CONFIG\_LANENUM  \_ACC\_EP | CONFIG\_LANENUM\_ACC\_EP\_1 | 8'd0 | 总是过渡 |

表14.48：配置

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No. | 先前状态 | 当前状态 | 跨因 | 转型原因 |
| 51 | CONFIG\_LANENUM  \_ACC\_EP\_1 | 配置\_完成 | 8'd0 | 如果接收到具有非PAD链路和非PAD通道编号的两个连续的TS2有序集合，  匹配所有非PAD链路和非PAD通道号的数据，下一个状态是配置完成（使用修改的TS） |
| 52 | CONFIG\_LANENUM\_WAIT\_EP | 8'd1 | 如果所有活动通道上都配置了通道号但未收到（使用修改后的TS） |
| 53 | TX\_ELEC\_IDLE\_ST | 8'd2 | 2个TS 1接收到PAD或24 ms超时计数器过期（使用修改后的TS） |
| 54 | 8'd3 | 2个TS 1接收到PAD或24 ms超时计数器过期（使用修改后的TS） |
| 55 | 配置\_完成 | 8'd4 | 如果接收到具有非PAD链路和非PAD通道编号的两个连续的TS2有序集合，  匹配所有非PAD链路和非PAD通道号的数据，下一个状态是配置。完成 |
| 56 | CONFIG\_LANENUM\_WAIT\_EP | 8'd5 | 如果所有活动通道 |
| 57 | TX\_ELEC\_IDLE\_ST | 8'd6 | 2个TS1接收到PAD或24 ms超时计数器过期 |
| 58 | 8'd7 | 2个TS1接收到PAD或24 ms超时计数器过期 |
| 59 | 配置\_完成 | CONFIG\_COMPLETE\_1 | 8'd0 | 总是过渡 |
| 60 | CONFIG\_COMPLETE\_1 | CONFIG\_COMPLETE\_2 | 8'd0 | 接收到有效的修改的TS 2（使用修改的TS） |
| 61 | TX\_ELEC\_IDLE\_ST | 8'd1 | 2ms超时计数器过期，gen3空闲到rlock转换为FFh（使用修改的TS） |
| 62 | CONFIG\_IDLE | 8'd2 | 2ms超时计数器过期，且tx\_state = TX\_COM（使用修改后的TS） |
| 63 | CONFIG\_COMPLETE\_2 | 8'd3 | 收到有效的修改TS2 |
| 64 | TX\_ELEC\_IDLE\_ST | 8'd4 | 2ms超时计数器过期，gen3空闲到rlock转换为FFh |
| 65 | CONFIG\_IDLE | 8'd5 | 2ms超时计数器已过期且tx\_state = TX\_COM |

表14.48：配置

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No. | 先前状态 | 当前状态 | 跨因 | 转型原因 |
| 66 | CONFIG\_COMPLETE\_2 | 检测\_安静 | 8'd0 | 发生CXL重定时器不匹配或发生CXL灵活总线使能位不匹配（使用修改的TS） |
| 67 | CONFIG\_IDLE | 8'd1 | CXL重定时器和灵活总线使能对齐，接收到8个无错误的修改TS（使用修改TS） |
| 68 | TX\_ELEC\_IDLE\_ST | 8'd2 | 2ms超时计数器过期，gen3空闲到rlock转换为FFh（使用修改的TS） |
| 69 | CONFIG\_IDLE | 8'd3 | 2ms超时计数器过期，且tx\_state = TX\_COM（使用修改后的TS） |
| 70 | 8'd4 | 收到8个无错误TS2 |
| 71 | TX\_ELEC\_IDLE\_ST | 8'd5 | 2 ms超时计数器过期，gen3空闲转换为rlock |
| 72 | CONFIG\_IDLE | 8'd6 | 2ms超时计数器已过期且tx\_state = TX\_COM |
| 73 | CONFIG\_IDLE | CONFIG\_IDLE\_1 | 8'd0 | Tx TS最后周期或Tx跳过最后周期 |
| 74 | TX\_ELEC\_IDLE\_ST | 8'd1 | 2ms超时计数器过期，gen3空闲到rlock转换为FFh |
| 75 | 恢复\_RCVR\_恢复 | 8'd2 | 如果idle\_to\_rlock\_transitioned变量小于FFh，则下一个状态为Recovery. RcvrLock。 |
| 76 | CONFIG\_IDLE\_1 | L0状态 | 8'd0 | 启用CXL模式并获取 |
| 77 | TX\_ELEC\_IDLE\_ST | 8'd1 | 2ms超时计数器过期，gen3空闲到rlock转换为FFh |
| 78 | 8'd2 | 2ms超时计数器过期，gen3空闲到rlock转换为FFh |
| 79 | 恢复\_RCVR\_恢复 | 8'd3 | 如果idle\_to\_rlock\_transitioned变量小于FFh，则下一个状态为Recovery. RcvrLock。 |

表14.49：L0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No. | 上一个LTSSM状态 | 当前LTSSM状态 | LTSSM  调试原因 | 描述 |
| 1 | L0\_STATE（7'd41） | RECOVERY\_RCVR\_ESTA（7'd42） | 0 | RP从CFG“重新训练链路” |
| 2 | L0状态 | 恢复\_RCVR\_恢复 | 1 | EP启动的速度变化再培训链接 |
| 3 | L0状态 | 恢复\_RCVR\_恢复 | 2 | EP 8GT请求EQ重新训练链接 |
| 4 | L0状态 | 恢复\_RCVR\_恢复 | 3 | EP 16 GT请求EQ重新训练链接 |
| 5 | L0状态 | 恢复\_RCVR\_恢复 | 4 | EP 32 GT请求EQ重新训练链接 |
| 6 | L0状态 | 恢复\_RCVR\_恢复 | 5 | RP发起的禁用链路请求 |
| 7 | L0状态 | 恢复\_RCVR\_恢复 | 6 | RP发起的热复位请求 |
| 8 | L0状态 | 恢复\_RCVR\_恢复 | 7 | RP发起的环回请求 |
| 9 | L0状态 | 恢复\_RCVR\_恢复 | 8 | NFTS |
| 10 | L0状态 | 恢复\_RCVR\_恢复 | 9 | RP自动均衡速度更改为Gen3及更高版本 |
| 11 | L0状态 | 恢复\_RCVR\_恢复 | 10 | RP自动速度更改为Gen1 |

表14.49：L0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No. | 上一个LTSSM状态 | 当前LTSSM状态 | LTSSM  原因 | 描述 |
| 12 | L0状态 | L1\_入口（7'd96） | 11 | 上层启动的L1入口 |
| 13 | L0状态 | L2\_ENTRY（7'd103） | 12 | 上层启动的L2条目 |
| 14 | L0状态 | 恢复\_RCVR\_恢复 | 13 | 错误情况：检测到RX ELEC IDLE，但没有EIOS |
| 15 | L0状态 | 恢复\_RCVR\_恢复 | 14 | 错误案例：PIPE\_RX\_VALID在L0中取消断言 |
| 16 | L0状态 | 恢复\_RCVR\_恢复 | 15 | TS1在L0状态 |
| 17 | L0状态 | 恢复\_RCVR\_恢复 | 16 | DL重放翻转重训练链接请求 |
| 18 | L0状态 | 恢复\_RCVR\_恢复 | 17 | 成帧错误重新训练链接 |
| 19 | L0状态 | 恢复\_RCVR\_恢复 | 18 | FC中继链路 |
| 20 | L0状态 | 恢复\_RCVR\_恢复 | 19 | EP/RP链路升级配置重新训练链路 |
| 21 | L0状态 | L0S\_ENTRY（7'd58） | 20 | 由于Tx不活动而 |
| 22 | L0状态 | 恢复\_RCVR\_恢复 | 21 | LM启动IORECAL请求重新训练链接 |
| 23 | L0状态 | 恢复\_RCVR\_恢复 | 22 | PHY发起的IORecal请求 |

表14.50：环回

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No. | 先前状态 | 当前状态 | 跨因 | 转型原因 |
| 1 | LOOPBACK\_从入口 | 循环\_SLAVE\_GEN2\_1 | 8'd0 | 当前速度不等于当前速率 |
| 2 | 循环\_SLAVE\_GEN2\_4 | 8'd1 | 当前速度等于当前速率 |
| 3 | LOOPBACK\_从机  \_输入\_从\_恢复 | 回送从机活动 | 8'd0 | 当在SDS期间断言tx\_ts\_last\_cycle时，等待SDS在Gen3中完全传输或Gen1、Gen2中的空闲符号，当end\_sds\_ordered\_set==1时实现24个周期的超时，当send\_sds\_ordered\_set==0时 |
| 4 | 循环\_SLAVE\_GEN2\_4 | 8'd1 | tx\_state = TX\_COM或TX\_EIEOS |
| 5 | 循环\_SLAVE\_GEN2\_1 | 循环\_SLAVE\_GEN2\_2 | 8'd0 | 第二个空闲序列已发送 |
| 6 | 循环\_SLAVE\_GEN2\_2 | 循环\_SLAVE\_GEN2\_3 | 8'd0 | 总是过渡 |
| 7 | 循环\_SLAVE\_GEN2\_3 | 回送从机活动 | 8'd0 | 下一个状态是环回。活动，如果符合接收位的TS1有序集，直接从这个状态被断言。从设备的发射机不需要在任何边界上转换为发送回送数据，并且允许截断正在进行的任何有序集。 |
| 8 | 循环\_SLAVE\_GEN2\_4 | 8'd1 | tx\_state = TX\_COM或TX\_EIEOS |

表14.50：环回

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No. | 先前状态 | 当前状态 | 跨因 | 转型原因 |
| 9 | 循环\_SLAVE\_GEN2\_4 | 恢复均衡  \_PHASE0 | 8'd0 | 下一个状态是Recovery.Equalization if Loopback.Entry is entered from Configuration- tion.Linkwidth.Start，最高公共数据速率为32.0 GT/s，将从站引导至该状态的TS 1有序集的增强型链路带宽或控制位为01 b。  - perform\_equalization\_for\_parameters变量设置为1b。- transmit\_modified  如果在将从机引导至该状态的TS 1有序集中，回送位中的传输修改的符合性模式被设置为1b，则\_compliance\_pattern\_in\_parameter变量被设置为1b。 |
| 10 | 回送从机活动 | 8'd1 | 如果环回。条目是从恢复输入的。均衡后，下一个状态是环回。在被测通道接收到两个连续的TS1有序集（环回位置位）后，激活。 |
| 11 | 8'd2 | TS1已收到。 |
| 12 | 循环\_SLAVE\_GEN2\_5 | 8'd3 | ts1\_eq\_received\_2\_with\_non\_zero\_ec\_all[3：2]！='b0 |
| 13 | 回送从机活动 | 8'd4 | 接收到的比特集 |
| 14 | 循环\_SLAVE\_GEN2\_5 | 回送从机活动 | 8'd0 | 总是过渡 |
| 15 | 回送从机活动 | LOOPBACK\_SLAVE\_EXIT | 8'd0 | 4检测到 |
| 16 | 8'd1 | 速度gen1和检测到的 |
| 17 | 8'd2 | 转速gen1和电气怠速推断 |
| 18 | LOOPBACK\_SLAVE\_EXIT | LOOPP\_SLAVE\_EXIT\_1 | 8'd0 | 接收到有效的Tx数据 |
| 19 | LOOPP\_SLAVE\_EXIT\_1 | TX\_ELEC\_IDLE\_ST | 8'd0 | 在关闭电源后，在一个周期内断言TX电气空闲后，2ms超时计数器过期 |
| 20 | LOOPBACK\_MASTER\_ENTRY | LOOPBACK\_MASTER\_EN-TRY\_1 | 8'd0 | 需要将当前速度切换到环回速率 |
| 21 | LOOPBACK\_MASTER\_EN-TRY\_5 | 8'd1 | 无需速度开关 |
| 22 | LOOPBACK\_MASTER\_EN-TRY\_1 | LOOPBACK\_MASTER\_EN-TRY\_2 | 8'd0 | 等待发送16个TS1。 |
| 23 | LOOPBACK\_MASTER\_EN-TRY\_2 | LOOPBACK\_MASTER\_EN-TRY\_3 | 8'd0 | 第二个空闲序列已发送 |
| 24 | LOOPBACK\_MASTER\_EN-TRY\_3 | LOOPBACK\_MASTER\_EN-TRY\_4 | 8'd0 | 总是过渡 |
| 25 | LOOPBACK\_MASTER\_EN-TRY\_4 | LOOPBACK\_MASTER\_EN-TRY\_5 | 8'd0 | 将当前速度更改为速率后，1ms超时计数器过期 |

表14.50：环回

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No. | 先前状态 | 当前状态 | 跨因 | 转型原因 |
| 26 | LOOPBACK\_MASTER\_EN-TRY\_5 | 恢复均衡  \_PHASE0 | 8'd0 | 下一个状态是恢复。均衡，如果数据速率被更改为32.0 GT/s和16 concentive TS 1有序集被发送与增强链路行为控制位设置为01 B的任何通道。perform\_equalization\_ for\_parameters变量设置为1b。 |
| 27 | LOOPBACK\_MASTER\_AC-TIVE | 8'd1 | 下一个状态是Loopback。如果Loopback.Entry是从Recovery输入的，则激活。均衡，并且被测通道接收两个连续的TS1有序集，其中回送位被置位。 |
| 28 | 8'd2 | 接收到2个TS，设置了 |
| 29 | LOOPBACK\_MASTER\_EXIT | 8'd3 | 48 ms超时计数器过期 |
| 30 | LOOPBACK\_MASTER  \_输入\_从\_恢复 | LOOPBACK\_MASTER\_EN-TRY\_5 | 8'd0 | TX TS最后周期或TX跳过最后周期 |
| 31 | LOOPBACK\_MASTER\_ACTIVE | LOOPBACK\_MASTER\_EXIT | 8'd0 | 已删除环回请求 |
| 32 | LOOPBACK\_MASTER\_EXIT | LOOPP\_MASTER\_EXIT\_1 | 8'd0 | 总是过渡 |
| 33 | LOOPP\_MASTER\_EXIT\_1 | LOOPP\_MASTER\_EXIT\_2 | 8'd0 | 已发送空闲序列 |
| 34 | LOOPP\_MASTER\_EXIT\_2 | TX\_ELEC\_IDLE\_ST | 8'd0 | 将SERDES驱动至电气怠速后，2ms超时计数器超时 |

表14.51：回收率

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No. | 先前状态 | 当前状态 | 跨因 | 转型原因 |
| 1 | 恢复\_RCVR\_恢复 | 恢复均衡阶段0 | 8'd0 | 速度为Gen3或更高，并使用预设设置 |
| 2 | RECOVERY\_RCVR\_LOCK\_1 | 8'd1 | 未设置 |

表14.51：回收率

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No. | 先前状态 | 当前状态 | 跨因 | 转型原因 |
| 3 | RECOVERY\_RCVR\_LOCK\_1 | 恢复均衡阶段0 | 8'd0 | 如果以8 G速度接收到8个连续的无误差EQ TS 1，速度变化位= 0，且EQ Control！00，过渡到恢复。均衡化。 |
| 4 | 恢复\_RCVR\_CFG | 8'd1 | 如果从L1进入Recovery，则在转换到Recovery之前等待T\_COMMON\_MODE。RcvrCfg |
| 5 | 8'd2 | 收到任何8个无错误TS有序集 |
| 6 | 恢复速度 | 8'd3 | Gen2变速恢复装置 |
| 7 | CONFIG\_LW\_START\_RC | 8'd4 | 如果directed\_speed\_change变量等于0 b并且接收到的TS 1或TS 2有序集上的speed\_change比特等于0 b，或者当前操作数据速率是2.5 GT/ s，并且2.5 GT/s数据速率是在发送的TS 1有序集和接收到的TS 1或TS 2有序集之间的最高的通常通告的数据速率，则下一状态是配置。根端口选定 |
| 8 | CONFIG\_LW\_START\_EP | 8'd5 | 如果directed\_ speed\_change变量等于0 b并且接收到的TS 1或TS 2有序集上的speed\_change比特等于0 b，或者当前操作的数据速率是2.5 GT/s，并且2.5 GT/s数据速率是发送的TS 1有序集和接收到的TS 1或TS 2有序集之间的最高共同通告数据速率，则下一状态是配置。已选择 |
| 9 | TX\_ELEC\_IDLE\_ST | 8'd6 | 未收到 |
| 10 | 恢复\_RCVR\_CFG | RECOVERY\_RCVR\_CFG\_1 | 8'd0 | 总是过渡 |
| 11 | 恢复\_RCVR\_CFG\_1 | 恢复速度 | 8'd0 | 收到任何8个无错误TS有序集 |
| 12 | 8'd1 | 接收到8个无差错TS，并设置了均衡请求 |
| 13 | 恢复\_空闲 | 8'd2 |  |
| 14 | CONFIG\_LW\_START\_RC | 8'd3 | 接收到8个TS 1，但链路或通道号不匹配，且选择了 |
| 15 | CONFIG\_LW\_START\_EP | 8'd4 | 接收到8个TS 1，链路或通道号不匹配，且选择了 |
| 16 | 恢复速度 | 8'd5 | 在恢复rcvr cfg状态 |
| 17 | 恢复\_空闲 | 8'd6 | 48 ms超时计数器过期，第3代空闲到rlock转换为FFh |
| 18 | TX\_ELEC\_IDLE\_ST | 8'd7 | 48 ms超时计数器过期 |

表14.51：回收率

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No. | 先前状态 | 当前状态 | 跨因 | 转型原因 |
| 19 | 恢复\_空闲 | CONFIG\_LW\_START\_RC | 8'd0 | 正在进行链路升级配置，并处于根端口模式 |
| 20 | CONFIG\_LW\_START\_EP | 8'd1 | 正在进行链路升级配置和端点模式 |
| 21 | 热门文章 | 8'd2 | 请求热复位 |
| 22 | DISABLE\_LINK | 8'd3 | 请求禁用链接 |
| 23 | LOOPBACK\_MASTER\_ENTRY\_FROM  \_恢复 | 8'd4 | 请求回送 |
| 24 | RECOVERY\_IDLE\_1 | 8'd5 | 转到恢复空闲1组 |
| 25 | DISABLE\_LINK | 8'd6 | 接收到断言链接禁用 |
| 26 | 热门文章 | 8'd7 | 收到任何2个请求，以断言热重置 |
| 27 | CONFIG\_LW\_START\_RC | 8'd8 | 收到任何2个有效TS1。根端口模式 |
| 28 | CONFIG\_LW\_START\_EP | 8'd9 | 收到任何2个有效TS1。端点模式 |
| 29 | LOOPBACK\_SLAVE\_ENTRY\_FROM\_RE-COVERY | 8'd10 | 接收到的比特集 |
| 30 | TX\_ELEC\_IDLE\_ST | 8'd11 | 2ms超时计数器过期，gen3空闲到rlock转换为FFh |
| 31 | 恢复\_RCVR\_恢复 | 8'd12 | 2ms超时计数器过期，第3代空闲到rlock的转换时间小于FFh |
| 32 | RECOVERY\_IDLE\_1 | L0状态 | 8'd0 | 空闲条件满足恢复和sds seq接收所有和CXL模式启用 |
| 33 | DISABLE\_LINK | 8'd1 | 收到禁用链接请求 |
| 34 | 热门文章 | 8'd2 | 收到热重置请求 |
| 35 | CONFIG\_LW\_START\_RC | 8'd3 | 收到任何2个有效TS1。根端口模式 |
| 36 | CONFIG\_LW\_START\_EP | 8'd4 | 收到任何2个有效TS1。端点模式 |
| 37 | LOOPBACK\_SLAVE\_ENTRY\_FROM\_RE-COVERY | 8'd5 | 请求回送 |
| 38 | TX\_ELEC\_IDLE\_ST | 8'd6 | 2ms超时计数器过期，gen3空闲到rlock转换为FFh |
| 39 | 恢复\_RCVR\_恢复 | 8'd7 | 2ms超时计数器过期，第3代空闲到rlock的转换时间小于FFh |
| 40 | 恢复  \_速度 | RECOVERY\_SPEED\_1 | 8'd0 | 总是过渡 |
| 41 | 恢复速度1 | RECOVERY\_SPEED\_2 | 8'd0 | 发送第二个空闲序列 |
| 42 | 恢复速度2 | RECOVERY\_SPEED\_3 | 8'd0 | Rx通道进入电气空闲 |
| 43 | TX\_ELEC\_IDLE\_ST | 8'd1 | 48 ms超时计数器在双通道进入电气空闲 |
| 44 | RECOVERY\_SPEED\_3 | 恢复\_RCVR\_恢复 | 8'd0 | 如果设置了start\_equalization，则接收link\_eq\_tx\_coefficients如果设置了no\_eq\_needed，则恢复no\_eq\_coefficients |
| 45 | TX\_ELEC\_IDLE\_ST | 8'd1 | 48 ms超时计数器过期。 |

表14.51：回收率

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No. | 先前状态 | 当前状态 | 跨因 | 转型原因 |
| 46 | 恢复均衡  \_PHASE0 | 恢复均衡阶段1 | 8'd0 | 根端口模式和速度为Gen5 |
| 47 | 8'd1 | 端点模式和TS 1 eq接收时的ec均为非零 |
| 48 | LOOPBACK\_SLAVE\_ENTRY\_FROM\_RE-COVERY | 8'd2 | 12 ms超时计数器过期，第5代执行了针对数据集的表单均衡 |
| 49 | 恢复速度 | 8'd3 | 12 ms超时计数器过期，未设置第5代每个表单的均衡 |
| 50 | 恢复均衡  \_PHASE1 | 恢复均衡阶段2\_1 | 8'd0 | 如果数据速率为32.0 GT/s且perform\_ equalization\_for\_decoder为0 b |
| 51 | LOOPBACK\_SLAVE\_ENTRY\_FROM\_RE-COVERY | 8'd1 | 下一个状态是环回。如果perform\_ equalization\_for\_equalization是1b并且测试中的通道接收八个连续的具有EC=00b的 |
| 52 | 恢复\_RCVR\_恢复 | 8'd2 | 环回位未设置 |
| 53 | LOOPBACK\_SLAVE\_ENTRY\_FROM\_RE-COVERY | 8'd3 | 下一个状态是Loopback。如果perform\_ equalization\_for\_allocation为1b且超时时间 |
| 54 | 恢复速度 | 8'd4 | 环回位未设置 |
| 55 | 恢复\_RCVR\_恢复 | 8'd5 | 下一个状态是Recovery.RcvrLock，如果所有配置通道接收到EC= 01 b的两个连续TS 1 Or- dered Set，则perform\_equaliza- tion\_for\_bandwidth为0 b，并且下游端口不想执行阶段2和阶段3。 |
| 56 | LOOPBACK\_MASTER\_ENTRY\_FROM\_RECOVERY | 8'd6 | 下一个状态是Loopback。如果perform\_ equalization\_for\_bandwidth是1b并且满足以下条件之一，则进入：被测通道接收EC=01b的两个连续TS1有序集，并且下游端口不想执行阶段2和阶段3。 |
| 57 | 恢复均衡阶段3\_1 | 8'd7 | 环回位未置位，均衡的阶段2和阶段3未被旁路 |
| 58 | LOOPBACK\_MASTER\_ENTRY\_FROM  \_恢复 | 8'd8 | 下一个状态是Loopback。如果perform\_ equalization\_for\_bandwidth是1b并且满足以下条件之一，则进入：24 ms超时计数器过期 |
| 59 | 恢复速度 | 8'd9 | 环回位未设置且24 ms超时计数器已过期 |

表14.51：回收率

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No. | 先前状态 | 当前状态 | 跨因 | 转型原因 |
| 60 | 恢复均衡阶段2\_1 | 恢复均衡阶段3\_1 | 8'd0 | 非下游端口 |
| 61 | LOOPBACK\_MASTER\_ENTRY\_FROM  \_恢复 | 8'd1 | 如果操作的数据速率是32.0 GT/s，则输入perform\_equaliza- tion\_for\_decoder是1b并且满足以下条件之一：a.被测通道在其最佳设置下操作，并且所有通道接收两个连续的TS 1有序集，其中接收到重定时器均衡扩展位设置为0b。 |
| 62 | 恢复\_RCVR\_恢复 | 8'd2 | 环回位未设置 |
| 63 | 恢复均衡阶段2\_2 | 8'd3 | 链路均衡未完成。 |
| 64 | LOOPBACK\_MASTER\_ENTRY\_FROM\_RECOVERY | 8'd4 | 如果操作的数据速率是32.0 GT/s，则输入perform\_equaliza- tion\_for\_decoder是1b并且满足以下条件之一：a. 24 ms超时，容差为-0 ms和+2 ms。根端口模式 |
| 65 | LOOPBACK\_SLAVE\_ENTRY\_FROM\_RE-COVERY | 8'd5 | 如果操作的数据速率是32.0 GT/s，则输入perform\_equaliza- tion\_for\_decoder是1b并且满足以下条件之一：a. 24 ms超时，容差为-0 ms和+2 ms。端点模式 |
| 66 | 恢复速度 | 8'd6 | 环回位未设置 |
| 67 | 恢复均衡  第二阶段 | 恢复均衡阶段2\_1 | 8'd0 | 等待远程PHY改变其系数 |
| 68 | 8'd1 | 等待2ms，以接收阶段2中的 |
| 69 | LOOPBACK\_MASTER\_ENTRY\_FROM\_RECOVERY | 8'd2 | 如果操作的数据速率是32.0 GT/s，则输入perform\_equaliza- tion\_for\_decoder是1b并且满足以下条件之一：a. 24 ms超时，容差为-0 ms和+2 ms。根端口模式 |
| 70 | LOOPBACK\_SLAVE\_ENTRY\_FROM\_RE-COVERY | 8'd3 | 如果操作的数据速率是32.0 GT/s，则输入perform\_equaliza- tion\_for\_decoder是1b并且满足以下条件之一：a. 24 ms超时，容差为-0 ms和+2 ms。端点模式 |
| 71 | 恢复速度 | 8'd4 | 环回位未设置 |

表14.51：回收率

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No. | 先前状态 | 当前状态 | 跨因 | 转型原因 |
| 72 | 恢复均衡  第三阶段 | 恢复均衡阶段3\_2 | 8'd0 | link\_eq\_intfc\_ack\_received = 1'b1 |
| 73 | LOOPBACK\_SLAVE\_ENTRY\_FROM\_RE-COVERY | 8'd1 | 下一个状态是Loopback。如果perform\_ equalization\_for\_bandwidth是1b并且满足以下条件之一，则进入：测试通道接收两个连续的TS 1有序集，EC=00b。 |
| 74 | 恢复\_RCVR\_恢复 | 8'd2 | 环回位未设置 |
| 75 | 恢复均衡阶段2\_1 | 8'd3 | 在ec all和根端口模式 |
| 76 | LOOPBACK\_MASTER\_ENTRY\_FROM  \_恢复 | 8'd4 | 下一个状态是Loopback。如果perform\_ equalization\_for\_bandwidth是1b并且满足以下条件之一，则进入：超时为32 ms，容差为-0 ms和+4 ms。根端口模式 |
| 77 | LOOPBACK\_SLAVE\_ENTRY\_FROM\_RE-COVERY | 8'd5 | 下一个状态是Loopback。如果perform\_ equalization\_for\_bandwidth是1b并且满足以下条件之一，则进入：超时为32 ms，容差为-0 ms和+4 ms。端点模式 |
| 78 | 恢复速度 | 8'd6 | 环回位未设置，32 ms超时计数器过期 |
| 79 | 恢复均衡  第三阶段 | 恢复均衡阶段3\_1 | 8'd0 | LTSSM转换为1'b0 |

表14.52：低功率

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No. | 先前状态 | 当前状态 | 跨因 | 转型原因 |
| 1 | L0S\_入口 | L0S\_1 | 8'd0 | 训练序列授予 |
| 2 | L0S\_1 | L0S\_2 | 8'd0 | 发送第二个空闲序列 |
| 3 | L0S\_2 | L0S\_3 | 8'd0 | 总是过渡 |
| 4 | L0S\_3 | L0S\_4 | 8'd0 | Tx ts周期计数大于定义的L0s电气空闲延迟 |
| 5 | L0S\_4 | L0S\_5 | 8'd0 | 确认物理状态 |
| 6 | L0S\_5 | 等待链接德克萨斯州 | 8'd0 | Tx ts周期计数大于定义的L0s最小延迟。 |
| 7 | 等待链接发送 | TX\_FTS\_输入 | 8'd0 | 如果检测到链路tx激活或接收到进入L1或L2的请求或请求重新训练 |
| 8 | TX\_FTS\_EN-尝试 | TX\_FTS\_1 | 8'd0 | tx\_state！= TX\_TS\_ID\_SEND\_SKIP\_GEN3\_2 |
| 9 | TX\_FTS\_1 | TX\_FTS\_2 | 8'd0 | 如果速度低于Gen3且tx\_state = TX\_TS\_ID  \_发送\_跳过 |
| 10 | 8'd1 | 如果速度Gen3或更高，且tx\_state = TX\_SDS |
| 11 | TX\_FTS\_2 | L0状态 | 8'd0 | 如果速度低于Gen3且tx\_state = TX\_IDLE |
| 12 |  | 8'd1 | 如果速度为Gen3 |
| 13 | L1入口 | L1\_1 | 8'd0 | 训练序列授予 |
| 14 | L1\_1 | L1\_2 | 8'd0 | 发送第二个空闲序列 |

表14.52：低功率

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No. | 先前状态 | 当前状态 | 跨因 | 转型原因 |
| 15 | L1\_2 | L1\_3 | 8'd0 | 总是过渡 |
| 16 | L1\_3 | L1\_4 | 8'd0 | 当速度低于Gen3时，收到EIOS并延迟2个时钟以退出 |
| 17 | 8'd1 | 收到EIOS，速度为Gen3或更高 |
| 18 | 8'd2 | 内部管道宽度不是32或64或128 |
| 19 | 恢复\_RCVR\_恢复 | 8'd3 | 在未接收EIOS的情况下 |
| 20 | 8'd4 | 如果链路在等待EIOS时重新训练，请移至恢复。 |
| 21 | L1\_4 | L1\_IDLE | 8'd0 | Tx ts周期计数大于定义的L1电气空闲延迟 |
| 22 | L1\_IDLE | L1\_退出 | 8'd0 | 如果L1.x尚未启动，请立即退出如果已启动，等待其达到退出状态在非Gen1速度下至少等待40 ns |
| 23 | L1\_退出 | 恢复\_RCVR\_恢复 | 8'd0 | 确认物理状态 |
| 24 | L2入口 | L2\_1 | 8'd0 | 训练序列授予 |
| 25 | L2\_1 | L2\_2 | 8'd0 | 发送第二个空闲序列 |
| 26 | L2\_2 | L2\_3 | 8'd0 | 总是过渡 |
| 27 | L2\_3 | L2\_4 | 8'd0 | 收到EIOS或电气空闲 |
| 28 | L2\_4 | L2\_IDLE | 8'd0 | 在进入L2\_IDLE |
| 29 | L2\_IDLE | 检测安静进入 | 8'd0 | Tx ts周期计数大于L2电气空闲延迟定义或未在任何活动通道上接收到电气空闲。 |
| 30 | 8'd1 |  |

表14.53：禁用、热复位和其他

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No. | 先前状态 | 当前状态 | 跨因 | 转型原因 |
| 1 | TX\_ELEC\_IDLE\_ST | TX\_ELEC\_IDLE\_1 | 8'd0 | 总是过渡 |
| 2 | TX\_ELEC\_IDLE\_1 | TX\_ELEC\_IDLE\_2 | 8'd0 | 发送第二个空闲序列 |
| 3 | TX\_ELEC\_IDLE\_2 | TX\_ELEC\_IDLE\_3 | 8'd0 | 当前速度高于Gen1 |
| 4 | 检测安静进入 | 8'd1 | 当前速度Gen1和tx空闲计数大于2 |
| 5 | TX\_ELEC\_IDLE\_3 | 检测\_安静\_开始-尝试 | 8'd0 | 等待恢复速度电气怠速 |
| 6 | DISABLE\_LINK | DISABLE\_LINK\_1 | 8'd0 | 总是过渡 |
| 7 | DISABLE\_LINK\_1 | DISABLE\_LINK\_2 | 8'd0 | 总是过渡 |
| 8 | DISABLE\_LINK\_2 | DISABLE\_LINK\_3 | 8'd0 | tx\_state = TX\_COM |
| 9 | DISABLE\_LINK\_3 | DISABLE\_LINK\_4 | 8'd0 | Tx有序集计数超过24 |
| 10 | DISABLE\_LINK\_4 | DISABLE\_LINK\_5 | 8'd0 | 速度为Gen2，发送 |
| 11 | 8'd1 | 速度不是Gen2 |

表14.53：禁用、热复位和其他

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No. | 先前状态 | 当前状态 | 跨因 | 转型原因 |
| 12 | DISABLE\_LINK\_5 | DISABLE\_LINK\_6 | 8'd0 | Tx空闲计数大于16，并且在活动通道上接收到电气空闲。端点模式 |
| 13 | 8'd1 | 在通道上接收到电气空闲之前，2ms超时计数器超时端点模式 |
| 14 | 8'd2 | 请求禁用链接 |
| 15 | DISABLE\_LINK\_6 | 检测安静进入 | 8'd0 | 当前速度为Gen1 |
| 16 | TX\_ELEC\_IDLE\_3 | 8'd1 | 当前速度高于Gen1 |
| 17 | 热门文章 | HOT\_RESET\_1 | 8'd0 | tx\_state = TX\_COM |
| 18 | HOT\_RESET\_1 | HOT\_RESET\_2 | 8'd0 | 根端口模式。热复位有效。 |
| 19 | TX\_ELEC\_IDLE\_ST | 8'd1 | 根端口模式和2ms超时计数器过期 |
| 20 | 8'd2 | 端点模式和2ms超时计数器过期。 |
| 21 | HOT\_RESET\_2 | TX\_ELEC\_IDLE\_ST | 8'd0 | 热复位请求解除断言 |

LTSSM状态编码

* + - 1. 外部LTSSM状态编码

下表提供了控制器的*dbg\_ltssm\_state[5：0]*输出上可见的LTSSM状态的编码以及从物理层UART状态寄存器0读取的LTSSM状态。

表14.54：外部LTSSM状态编码

|  |  |
| --- | --- |
| 外部LTSSM状态编码 | 状态值[5：0] |
| LTSSM\_检测\_安静 | 6点 |
| LTSSM\_检测\_活动 | 6'h1 |
| LTSSM\_轮询\_活动 | 6'h2 |
| LTSSM\_轮询\_依从性 | 6'h3 |
| LTSSM\_轮询\_配置 | 6'h4 |
| LTSSM\_CONFIGURATION\_LINK WIDTH\_START | 6'h5 |
| LTSSM\_CONFIGURATION\_LINK WIDTH\_ACCEPT | 6'h6 |
| LTSSM\_CONFIGURATION\_LANENUM\_ACCEPT | 6'h7 |
| LTSSM\_CONFIGURATION\_LANENUM\_WAIT | 6'h8 |
| LTSSM\_CONFIGURATION\_COMPUTER | 6'h9 |
| LTSSM\_CONFIGURATION\_IDLE | 6 '哈 |
| LTSSM\_RECOVERY\_RCVR\_REPORT | 6'HB |
| LTSSM\_RECOVERY\_SPEED | 6'HC |
| LTSSM\_RECOVERY\_RCVR\_CFG | 6 'HD |
| LTSSM\_恢复\_空闲 | 6 '他 |
| LTSSM\_L0\_STATE | 6'h10 |
| LTSSM\_RX\_L0S\_ENTRY | 6'h11 |
| LTSSM\_RX\_L0S\_IDLE | 6'h12 |
| LTSSM\_RX\_L0S\_FTS | 6'h13 |

表14.54：外部LTSSM状态编码

|  |  |
| --- | --- |
| 外部LTSSM状态编码 | 状态值[5：0] |
| LTSSM\_TX\_L0S\_ENTRY | 6'h14 |
| LTSSM\_TX\_L0S\_IDLE | 6'h15 |
| LTSSM\_TX\_L0S\_FTS | 6'h16 |
| LTSSM\_L1\_ENTRY | 6'h17 |
| LTSSM\_L1\_IDLE | 6'h18 |
| LTSSM\_L2\_IDLE | 6'h19 |
| LTSSM\_L2\_TRANSMIT\_WAKE | 6'h1a |
| LTSSM\_已禁用 | 6'h20 |
| LTSSM\_LOOPBACK\_ENTRY\_MASTER | 6'h21 |
| LTSSM\_LOOPBACK\_ACTIVE\_MASTER | 6'h22 |
| LTSSM\_LOOPBACK\_EXIT\_MASTER | 6'h23 |
| LTSSM\_LOOPBACK\_ENTRY\_SLAVE | 6'h24 |
| LTSSM\_LOOPBACK\_ACTIVE\_SLAVE | 6'h25 |
| LTSSM\_LOOPBACK\_EXIT\_SLAVE | 6 'h26 |
| LTSSM\_HOT\_ | 6'h27 |
| LTSSM\_RECOVERY\_EQUALIZATION\_PHASE0 | 6'h28 |
| LTSSM\_RECOVERY\_EQUALIZATION\_PHASE1 | 6'h29 |
| LTSSM\_RECOVERY\_EQUALIZATION\_PHASE2 | 6'h2A |
| LTSSM\_RECOVERY\_EQUALIZATION\_PHASE3 | 6'h2B |

* + - 1. 内部LTSSM状态编码

下表提供了控制器的*dbg\_debug\_data\_out[9：3]*输出上可见的内部LTSSM状态的编码

表14.55：内部LTSSM状态编码

|  |  |
| --- | --- |
| 内部LTSSM状态编码 | 状态值[6：0] |
| 检测\_安静 | 7'd0 |
| 检测安静进入 | 7'd1 |
| 检测\_活动 | 7'd2 |
| 检测\_ACTIVE\_1 | 7'd3 |
| 检测\_ACTIVE\_2 | 7'd4 |
| 检测\_ACTIVE\_3 | 7'd5 |
| RCVR\_DETECTED\_ST | 7'd6 |
| RCVR\_DETECTED\_1 | 7'd7 |
| 轮询\_活动 | 7'd8 |
| 轮询\_ACTIVE\_1 | 7'd9 |
| 轮询\_ACTIVE\_2 | 7'd10 |
| 轮询\_ACTIVE\_3 | 7'd11 |
| 轮询\_依从性 | 7'd12 |

表14.55：内部LTSSM状态编码

|  |  |
| --- | --- |
| 内部LTSSM状态编码 | 状态值[6：0] |
| 投票\_COMPANCE\_1 | 7'd13 |
| 轮询配置 | 7'd14 |
| POLLING\_CONFIG\_1 | 7'd15 |
| POLLING\_CONFIG\_2 | 7'd16 |
| CONFIG\_LW\_START\_RC | 7'd17 |
| CONFIG\_LW\_START\_RC\_1 | 7'd18 |
| CONFIG\_LW\_START\_RC\_2 | 7'd19 |
| 配置\_LW\_ACC\_RC | 7'd20 |
| CONFIG\_LANENUM\_WAIT\_RC | 7'd21 |
| CONFIG\_LANENUM\_WAIT\_RC\_1 | 7'd22 |
| CONFIG\_LANENUM\_ACC\_RC | 7'd23 |
| CONFIG\_LW\_START\_EP | 7'd24 |
| CONFIG\_LW\_START\_EP\_1 | 7 'd25 |
| CONFIG\_LW\_START\_EP\_2 | 7'd26 |
| CONFIG\_LW\_ACC\_EP | 7'd27 |
| CONFIG\_LANENUM\_WAIT\_EP | 7'd28 |
| CONFIG\_LANENUM\_WAIT\_EP\_1 | 7'd29 |
| CONFIG\_LANENUM\_ACC\_EP | 7'd30 |
| CONFIG\_LANENUM\_ACC\_EP\_1 | 7'd31 |
| 虚拟状态1 | 7'd32 |
| 配置\_完成 | 7'd33 |
| CONFIG\_COMPLETE\_1 | 7'd34 |
| CONFIG\_COMPLETE\_2 | 7'd35 |
| CONFIG\_IDLE | 7'd36 |
| CONFIG\_IDLE\_1 | 7'd37 |
| 虚拟状态2 | 7'd38 |
| 虚拟状态3 | 7'd39 |
| 虚拟状态\_4 | 7'd40 |
| L0状态 | 7'd41 |
| 恢复\_RCVR\_恢复 | 7'd42 |
| RECOVERY\_RCVR\_LOCK\_1 | 7'd43 |
| 恢复\_RCVR\_CFG | 7'd44 |
| RECOVERY\_RCVR\_CFG\_1 | 7'd45 |
| 恢复\_空闲 | 7'd46 |
| RECOVERY\_IDLE\_1 | 7'd47 |
| DISABLE\_LINK | 7'd48 |
| DISABLE\_LINK\_1 | 7'd49 |
| DISABLE\_LINK\_2 | 7'd50 |
| DISABLE\_LINK\_3 | 7'd51 |

表14.55：内部LTSSM状态编码

|  |  |
| --- | --- |
| 内部LTSSM状态编码 | 状态值[6：0] |
| DISABLE\_LINK\_4 | 7'd52 |
| DISABLE\_LINK\_5 | 7'd53 |
| DISABLE\_LINK\_6 | 7'd54 |
| DISABLE\_LINK\_7 | 7'd55 |
| 热门文章 | 7'd56 |
| HOT\_RESET\_1 | 7'd57 |
| HOT\_RESET\_2 | 7'd58 |
| HOT\_RESET\_3 | 7'd59 |
| L0S\_入口 | 7'd60 |
| L0S\_1 | 7'd61 |
| L0S\_2 | 7'd62 |
| L0S\_3 | 7'd63 |
| L0S\_4 | 7'd64 |
| L0S\_5 | 7'd65 |
| 等待链接德克萨斯州 | 7'd66 |
| TX\_FTS\_输入 | 7'd67 |
| TX\_FTS\_1 | 7'd68 |
| TX\_FTS\_2 | 7'd69 |
| TX\_ELEC\_IDLE\_ST | 7'd70 |
| TX\_ELEC\_IDLE\_1 | 7'd71 |
| TX\_ELEC\_IDLE\_2 | 7'd72 |
| TX\_ELEC\_IDLE\_3 | 7'd73 |
| 恢复速度 | 7'd74 |
| RECOVERY\_SPEED\_1 | 7'd75 |
| RECOVERY\_SPEED\_2 | 7'd76 |
| RECOVERY\_SPEED\_3 | 7'd77 |
| POLLING\_COMPANCE\_GEN23 | 7'd78 |
| PolLING\_COMPANCE\_GEN23\_1 | 7'd79 |
| PolLING\_COMPANCE\_GEN23\_2 | 7'd80 |
| PolLING\_COMPANCE\_GEN23\_3 | 7'd81 |
| PolLING\_COMPANCE\_GEN23\_4 | 7'd82 |
| 投票\_COMPLIANCE\_GEN23\_5 | 7'd83 |
| PolLING\_COMPANCE\_GEN23\_6 | 7'd84 |
| PolLING\_COMPANCE\_GEN23\_7 | 7'd85 |
| PolLING\_COMPANCE\_GEN23\_8 | 7'd86 |
| LOOPBACK\_Slave\_ENTRY | 7'd87 |
| LOOPBACK\_SLAVE\_ENTRY\_FROM\_RECOVERY | 7'd88 |
| LOOPP\_SLAVE\_EXIT\_1 | 7'd89 |

表14.55：内部LTSSM状态编码

|  |  |
| --- | --- |
| 内部LTSSM状态编码 | 状态值[6：0] |
| LOOPBACK\_SLAVE\_EXIT | 7'd90 |
| 循环\_SLAVE\_GEN2\_1 | 7'd91 |
| 循环\_SLAVE\_GEN2\_2 | 7'd92 |
| 循环\_SLAVE\_GEN2\_3 | 7'd93 |
| 循环\_SLAVE\_GEN2\_4 | 7'd94 |
| 循环\_SLAVE\_GEN2\_5 | 7'd95 |
| 回送从机活动 | 7'd96 |
| L1入口 | 7'd97 |
| L1\_1 | 7'd98 |
| L1\_2 | 7'd99 |
| L1\_3 | 7'd100 |
| L1\_4 | 7'd101 |
| L1\_IDLE | 7'd102 |
| L1\_退出 | 7'd103 |
| L2入口 | 7'd104 |
| L2\_1 | 7'd105 |
| L2\_2 | 7'd106 |
| L2\_3 | 7 'd107 |
| L2\_4 | 7'd108 |
| L2\_5 | 7'd109 |
| L2\_IDLE | 7 'd110 |
| LOOPBACK\_MASTER\_ENTRY | 7'd111 |
| 循环主条目1 | 7'd112 |
| 循环主条目2 | 7'd113 |
| 循环主条目3 | 7'd114 |
| 循环主条目4 | 7'd115 |
| 循环主条目5 | 7'd116 |
| LOOPBACK\_MASTER\_ENTRY\_FROM\_RECOVERY | 7'd117 |
| LOOPBACK\_MASTER\_ACTIVE | 7'd118 |
| LOOPBACK\_MASTER\_EXIT | 7'd119 |
| LOOPP\_MASTER\_EXIT\_1 | 7'd120 |
| LOOPP\_MASTER\_EXIT\_2 | 7'd121 |
| 恢复均衡阶段0 | 7'd122 |
| 恢复均衡阶段1 | 7 'd123 |
| 恢复均衡阶段2\_1 | 7'd124 |
| 恢复均衡阶段2\_2 | 7'd125 |
| 恢复均衡阶段3\_1 | 7'd126 |
| 恢复均衡阶段3\_2 | 7'd127 |

# 第15章：验证

第15章：**验证**

* 1. 验证环境

Cadence® PCIe® Verilog Delivery Testbench提供了一个简单的基于Verilog的测试环境，使您能够在子系统模式下运行时生成并观察通过PCIe控制器和PHY的入站和出站流量

它是一个独立的环境，不需要任何VIP（*验证IP*）。

* + - 与使用PCIe和ARM® AMBA**®** VIP的环境相比，此测试平台生成的流量和场景有限
    - Verilog交付测试台不用于验证控制器、PHY的功能或控制器与PHY的集成
  1. 验证概述

PCIe子系统（控制器和PHY）的测试台实例通过串行链路连接Verilog总线功能模型（BFM）连接到应用级接口- HLS/AXI主端口和HLS/AXI从端口以及APB从端口。

APB BFM连接到PCIe子系统内PCIe控制器、PHY和配置寄存器的配置端口。HLS/AXI BFM连接到PCIe控制器的应用程序接口SRAM代表IP使用的PCIe控制器存储器。

有关更多信息，请参阅PCIe*Verilog Delivery Testbench用户指南*。

AXI Maste BFM



vb2b\_tb

APB/AXI-

Lite Master BFM

AXI/HLS/rCLS

主BFM

RAMs

地址解码

PCIe PHY

参考IOS

DUT IOS

PCIe PHY

地址解码

APB/AXI-

Lite Master BFM

AXI/HLS

主BFM

RAMs

PCIe控制器

子系统配置寄存器

PCIe控制器

子系统配置寄存器

**图15.1：**HPA子系统测试台框图

 注：该图仅用于代表性目的，块根据所选配置适用。

# 第16章附录C：PCIe配置功能链接列表

第16章附录**C：PCIe配置功能链接列表**

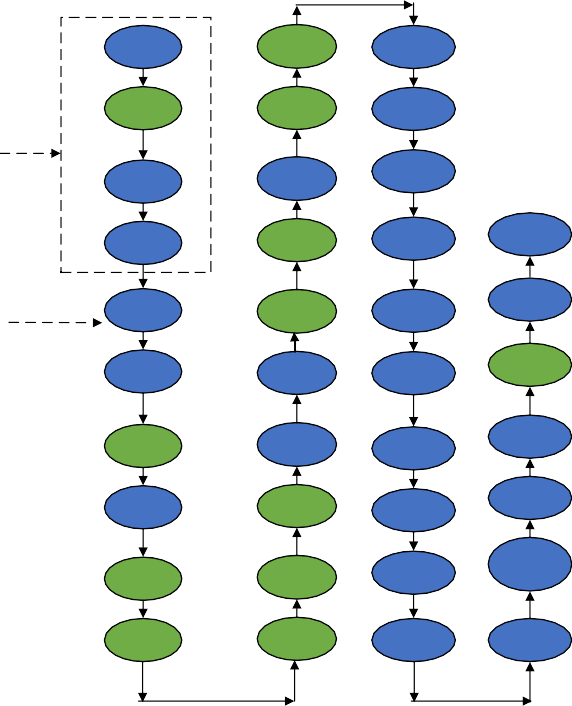
16.1 PCIe控制器功能链表实现

本节介绍PCIe控制器的PCIe配置功能链接列表。下表显示了PCIe控制器中所有功能的PCIe控制器功能链接列表实现。如果任何能力不存在或对所选控制器无效，则在此之前的能力将指向控制器中存在的下一个能力。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 能力名称 | 能力广告 | 能力下一点- er | 要跳过的 | 模式能力有效 |
| PCI PM能力 | 0x80 | 0x88 | - | EP RP |
| VPD能力 | 0x88 | 0x90 | - | EP |
| MSI能力 | 0x90 | 0xB0 | 未选择 | EP RP |
| MSIX能力 | 0xB0 | 0xC0 | 未选择 | EP RP |
| PCIE能力 | 0xC0 | 0x00（PCI Base结束） | - | EP RP |
| AER能力 | 0x100 | 0x144 | 第一个扩展帽，始终存在 | EP RP |
| ARI能力 | 0x144 | 0x150 | 未选择 | EP |
| 设备串行可扩展性 | 0x150 | 0x160 | 未选择 | EP RP |
| 功率放大能力 | 0x160 | 0x180 | 未选择 | EP |
| 可调整大小的BAR功能 | 0x180 | 0x1B8 | 未启用可 | EP |
| LTR能力 | 0x1B8 | 0x1C0 | 未选择 | EP |
| DPA功能 | 0x1C0 | 0x200 | 未选择 | EP |
| SRIOV能力 | 0x200 | 0x274 | num\_vfs为零 | EP |
| TPH能力 | 0x274 | 0x300 | 未选择 | EP RP |
| PCIE辅助分机容量 | 0x300 | 0x400 | strap\_pcie\_rate\_max 2 | EP RP |
| 供应商特定功能 | 0x400 | 0x440 | - | EP |
| PASID能力 | 0x440 | 0x4C0 | 未选择 | EP |
| VC能力 | 0x4C0 | 0x5C0 | num\_vcs == 1 | EP RP |
| ATS能力 | 0x5C0 | 0x640 | 未选择 | EP |
| ATS页面请求能力 | 0x640 | 0x900 | 未选择 | EP |
| L1 PM子状态能力 | 0x900 | 0x910 | 未选择 | EP RP |
| DL特性功能 | 0x910 | 0x920 | strap\_pcie\_rate\_max 3 | EP RP |
| RX裕度能力 | 0x920 | 0x9C0 | strap\_pcie\_rate\_max 3 | EP RP |
| PL 16.0 GT/s能力 | 0x9C0 | 0xA40 | strap\_pcie\_rate\_max 3 | EP RP |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 能力名称 | 能力广告 | 能力下一点- er | 要跳过的 | 模式能力有效 |
| PL 32.0 GT/s能力 | 0xA40 | 0xAA0 | strap\_pcie\_rate\_max 4 | EP RP |
| PL 64.0 GT/s能力 | 0xAA0 | 0xAD 0 | strap\_pcie\_rate\_max 5 | EP RP |
| Flit日志记录功能 | 0xAD 0 | 0xB10 | strap\_pcie\_rate\_max 5 | EP RP |
| 微片性能测量能力 | 0xB10 | 0xB40 | strap\_pcie\_rate\_max 5 | EP RP |
| 微片错误注入功能 | 0xB40 | 0xA20 | strap\_pcie\_rate\_max 5 | EP RP |
| PTM能力 | 0xA20 | 0xA30 | 未选择 | EP RP |
| NPEM能力 | 0xA30 | 0xA70 | 未选择 | EP RP |
| 备用协议协商能力 | 0xA70 | 0xA90 | 未选择 | EP RP |
| 设备3能力 | 0xA90 | 0xB70 | 不支持延迟MWR，strap\_pcie\_rate\_max 5 | EP RP |
| IDE扩展功能 | 0xB70 | 0xDE0 | 未选择IDE或未选择PF 0 | EP RP |
| DOE扩展功能 | 0xDE0 | 0xE00 | 未选择IDE或未选择PF 0 | EP |
| 指定的供应商特定能力 | 0xE00 | 0x000 | 未选择CXL或未选择PF 0 | EP RP |

[*图16.1*](#_bookmark230)显示了PCIe控制器中所有功能的PCIe控制器功能链接列表实现



PCI PM 0x080

ATS页面请求帽0x640

L1 PMSUBSTATES CAP

0x900

PCI

兼容的配置空间功能。

VPD 0x088

ATS CAP 0x5C0

DL特征帽0x910

微星0x090

虚拟通道帽0x4C0

RX MARGINIG帽0x920

MSI-X 0x090

PASID帽0x440

PL 16.0 GT/s CAP 0x9C0

最后指针0x000

PCI Express兼容配置空间功能。

PCI CAP 0x0C0

供应商专用帽0x400

PL 32.0 GT/s CAP 0xA40

标记的供应商专用帽0xE00

AER 0x100

PCIe

辅助外部盖0x300

PL 64.0 GT/s CAP 0xAA0

DOE扩展帽

0xDE0

ARI 0x140

TPH帽0x274

FLIT记录帽0xAD 0

IDE扩展帽0xB70

器械序列号帽0x150

SRIOV CAP 0x200

FLIT性能测量帽

0xB10

器械3帽0xA90

电源预算上限

0x160

DPA CAP 0x1C0

FLIT错误注射帽

0xB40

替代方案谈判上限

0xA70

可调整杆帽0x180

LTR CAP 0x1B8

PTM CAP 0xA20

NPEM CAP 0xA30

**图16.1：**PCIe控制器功能链表实现

# 第17章：附录G

第17章：**附录G**

17.1已发布版本

下表显示了已发布版本的*Cadence Controller IP for PCIe 5.0 HPA用户指南*之间的差异。

|  |  |  |
| --- | --- | --- |
| 版本 | 发布日期 | 评论 |
| 0.7 | 2020年12月07日 | 初始版本 |
| 0.71 | 2021年3月3日 | 添加了AXI Master部分 |
| 0.72 | 2021年11月11 | 添加了偏斜组件附录 |
| 增加了通过PCIe链路访问IP寄存器 |
| 添加附录：PCIe配置功能链接列表 |
| 添加附录：LTSSM状态名称 |
| 增加综合战略章节 |
| 0.73 | 2022年3月8日 | 在第一章中，IP   * 更新概述部分 * 更新核心配置部分 |
| 在第章中，应用程序接口AXI4   * 更新AXI从机写操作部分 * 更新AXI从机读操作部分 * 更新了AXI从机（错误）响应部分下的主题   - 添加主题，链接断开事件   * AXI链路故障处理部分 * AXI Slave部分 |
| 在第一章中，IP寄存器接口   * 更新概述部分 |
| 更新附录C：PCIe配置功能链接列表 |
| 第二节综合战略   * 添加了标准电池更换 |
| 一般和编辑更新 |
| 0.74 |
| 0.75 | 2022年4月25日 | * 在部分中，请求转换   - 更新表，512位数据路径宽度的AWLEN和AWSIZE信号化示例。注册.   * 更新了AXI从机（错误）响应部分。 |

|  |  |  |
| --- | --- | --- |
| 版本 | 发布日期 | 评论 |
| 0.76 | 2022年7月7日 | * 在第章中，应用程序接口AXI4   + AXI从描述符通过边带部分   + AXI从机描述符示例（正常消息）部分   + AXI从机写操作部分   + AXI从机部分   + AXI从机访问说明部分   + AXI从机读操作部分   + 未过帐事务处理的标记管理 * 在其他接口中，更新了其他接口中的关系图 * 更新了关于此文档中的交付包信息 * 在电源管理中，在内部电源关闭部分下添加了主题 * PCIe控制器功能链表实现。 * 一般和编辑更新 |
| 0.77 | 2022年7月20日 | * 在“应用程序接口AXI4”一章中，更新了AXI区域基址寄存器 * 更新PCIe控制器功能链表实现。 |
| 0.78 | 2022年8月19日 | * 删除了AXL从寄存器，并提供了HTML源文件的参考。 |
| 0.79 | 2022年9月15日 | 在“应用程序接口AXI4”一章中，在AXI从机部分下添加了[*应用层读取大小*](#_bookmark64) |
| 1.01 | 2022年12月1日 | * 更新了[*表5.2*](#_bookmark38)中的addr[25：24]解码值。 * 更新了[*图5.1*](#_bookmark36) |
| 1.02 | 2023年1月24日 | * 更新重置序列。参见[*重置序列*](#_bookmark175) * 更新了ATS翻译请求支持的[*AXI下属访问说明*](#_bookmark78) |
| 1.03 | 2023年2月27日 | 更新了[*表8.6*](#_bookmark80)和[*表8.7*](#_bookmark81) |
| 1.04 | 2023年3月29日 | * 添加了[*重要产品数据（VPD）集成*](#_bookmark183)。 * 更新[*AXI下属（错误）响应*](#_bookmark83)。 * [*通过边带描述符*](#_bookmark76)更新AXI从属访问 |
| 1.05 | 2023年4月4日 | * 添加[*了总VF数量不均匀的示例*](#_bookmark69) * 添加[*了具有统一数量的总VF的示例*](#_bookmark68) |
| 1.06 | 2023年5月5日 | * 非技术更新- |
| 1.07 | 2023年6月2日 | * 更新[*表5.2*](#_bookmark38) * 更新[*Reg时钟使用*](#_bookmark27) |
| 1.08 | 2023年9月4日 | 更新了章节：综合和物理设计指南. |
| 1.09 | 2023年11月07日 | 章节重组。 |
| 更新了复位框图。参见[*重置参考块图*](#_bookmark177)  更新启动顺序。请参见热插拔部分和子部分中添加[*的启动顺序*](#_bookmark173)请参见和小节。 |